mathematics

Evolutionary Computation 2020

Edited by

Gai-Ge Wang and Amir H. Alavi

Printed Edition of the Special Issue Published in *Mathematics*

www.mdpi.com/journal/mathematics

MDPI

Evolutionary Computation 2020

Evolutionary Computation 2020

Editors

Gai-Ge Wang
Amir H. Alavi

MDPI • Basel • Beijing • Wuhan • Barcelona • Belgrade • Manchester • Tokyo • Cluj • Tianjin

Editors

Gai-Ge Wang
Ocean University of China
China

Amir H. Alavi
University of Pittsburgh
USA

Editorial Office
MDPI
St. Alban-Anlage 66
4052 Basel, Switzerland

This is a reprint of articles from the Special Issue published online in the open access journal *Mathematics* (ISSN 2227-7390) (available at: https://www.mdpi.com/journal/mathematics/special_issues/Evolutionary_Computation_2020).

For citation purposes, cite each article independently as indicated on the article page online and as indicated below:

LastName, A.A.; LastName, B.B.; LastName, C.C. Article Title. *Journal Name* **Year**, *Volume Number*, Page Range.

ISBN 978-3-0365-2394-1 (Hbk)
ISBN 978-3-0365-2395-8 (PDF)

Contents

About the Editors

Gai-Ge Wang is an associate professor at the Ocean University of China, China. His publications have been cited over 9600 times (Google Scholar). His latest Google h-index and i10-index are 52 and 106, respectively. Fifteen and sixty-six papers are selected as Highly Cited Papers by Web of Science, and Scopus (as of October 2021), respectively. He was selected as one of "2021 Highly Cited Researchers" by Clarivate. He was selected as one of "2020 Highly Cited Chinese Researchers" in computer science and technology by Elsevier. He was selected as World's Top 2% Scientists 2020, ranked 3840 in 2019, and ranked 88,554 in career-long citation impact. One of his papers was selected as for the "100 Most Influential International Academic Papers in China". Another of his papers ranks 1 in the selection of the latest high-impact publications in computer science by Chinese researchers from Springer Nature in 2019. He is a senior member of SAISE, SCIEI, a member of IEEE, IEEE CIS, and ISMOST. He served as Editorial Advisory Board Member of *Communications in Computational and Applied Mathematics* (*CCAM*), Associate Editor of *IJCISIM*, an Editorial Board Member of *IEEE/CAA Journal of Automatica Sinica*, *Mathematics*, *IJBIC*, *Karbala Int J of Modern Science*, and the *Journal of AIS*. He served as Guest Editor for many journals including *Mathematics*, *IJBIC*, *FGCS*, *Memetic Computing* and *Operational Research*. His research interests are swarm intelligence, evolutionary computation, and big data optimization.

Amir H. Alavi is an Assistant Professor in the Department of Civil and Environmental Engineering, and holds a courtesy appointment in the Department of Bioengineering at the University of Pittsburgh. Prior to joining Pitt, Dr. Alavi was an Assistant Professor of Civil Engineering at the University of Missouri. Dr. Alavi is the director of the Pitt's Intelligent Structural Monitoring and Response Testing (iSMaRT) Laboratory, which focuses on integrating sensing, energy harvesting, material, and computational technologies to build a new generation of multifunctional structural health monitoring systems. Dr. Alavi's multidisciplinary research involves advancing the knowledge and technology required to create multifunctional architected material and structural systems with self-diagnostic and self-powering capabilities for a broad range of sensing and monitoring applications. His research goal is to harness the power of these technologies enhanced by engineering informatics for tackling problems in the fields of civil infrastructure, construction, aerospace, and biomedical engineering. Dr. Alavi's original and seminal contributions to developing and deploying advanced machine learning and bio-inspired computational techniques have established a road map for their broad applications in various engineering domains. Dr. Alavi has authored 8 books and nearly 200 publications in archival journals, book chapters, and conference proceedings. He has received a number of award certificates for his journal articles. He is among the Google Scholar 200 Most Cited Authors in Civil Engineering, Web of Science ESI's World Top 1% Scientific Minds in 2018, and Stanford University list of Top 1% Scientists in the World in 2020.

Preface to ”Evolutionary Computation 2020”

Intelligent optimization is based on the mechanism of computational intelligence to refine a suitable feature model, design an effective optimization algorithm, and then obtain an optimal or satisfactory solution to a complex problem. Intelligent algorithms should try to ensure global optimization quality, fast optimization efficiency and robust optimization performance. Many researchers have made different findings on the study of intelligent optimization algorithms.

In terms of improving algorithm performanc, Li et al. proposed a co-evolutionary algorithm based on a dynamic learning strategy. The evolution is mainly achieved by using the Pareto criterion and the non-Pareto criterion for two populations, respectively, and using the information exchange between the two populations to better explore the whole target space. Hussien et al. proposed two binary variants of the Whale Optimization Algorithm (WOA), called bWOA-S and bWOA-V. They are used to reduce the complexity and improve the performance of the system by selecting important features for classification purposes. Sun et al. proposed a steering-based variational approach for solving the premature convergence problem of the success history-based adaptive differential evolution algorithm in a high-dimensional search space. Wei et al. optimized the particle swarm optimization algorithm by quantum behavior and optimized the krill swarm algorithm by simulated annealing, thus proposing a new hybrid algorithm called the annealed krill quantum particle swarm optimization (AKQPSO) algorithm. A comprehensive review of algorithms based on elephant grazing optimization and their applications is presented by Li et al. Various aspects of EHO variants for continuous optimization, combinatorial optimization, constrained optimization, and multi-objective optimization are reviewed. Future research directions in the field of EHO are further discussed. Novak et al. proposed a new metric for game feature verification in real-time strategy (RTS) games, comparing evolutionary and tree-based approaches for game feature verification in real-time strategy games.

In order to solve complex problems. Li et al. proposed a discrete artificial bee colony (DABC) algorithm based on similarity and non-dominated solution ordering, which can solve the fuzzy hybrid green shop scheduling problem with fuzzy processing time. Li et al. proposed a new modal strategy based on particle swarm optimization algorithm, which can solve the severe nonlinear problem in one-dimensional geodesic electromagnetic inversion. Zhao et al. proposed a novel multi-objective optimization solution, MooFuzz, which identifies different states of the seed pool and continuously collects different information about the seeds to guide seed scheduling and energy allocation. The method can be used to find bugs and vulnerabilities in software. Wang et al. proposed a new quantum-inspired differential evolution algorithm based on the gray wolf optimizer, which can solve the 0-1 backpacking problem. Zhang et al. proposed a pair-wise ant colony optimization algorithm combined in position-based learning in order to solve the traveling merchant problem (TSP). Two strategies for constructing opposite paths based on TSP solution features for OBL were also proposed. Muhammad et al. conducted a comparative study of multi-objective evolutionary algorithms and single-objective evolutionary algorithms in optimizing the knapsack problem (KNP) and the traveling merchant problem (TSP). Marrero et al. proposed a multi-objective modal approach based on the decomposition-based multi-objective evolutionary algorithm (MOEA/D). The method contains crossover operators specifically designed for this problem. In addition, an interim iterative local search (ILS) is considered in the improvement phase. The modified algorithm can be used to develop healthy, balanced and inexpensive menu plans. Rahnamayan et al. proposed a new and

improved Pareto dominance depth ranking strategy that uses some dominance indicators obtained from the basic Pareto dominance depth ranking and some ranked statistical indicators to rank scientific outcomes. Li et al. proposed a new CS extension with Q-learning steps and genetic operators, namely the dynamic step cuckoo search algorithm (DMQL-CS), which can be used to solve the logistics distribution center site selection problem.

The editors are confident that this book will help beginners to understand the principles and design of intelligent algorithms. This book serves as a viable resource for readers interested in the applications of intelligent algorithms. It will further promote the development and improvement of intelligent algorithm research, strengthen the research of computational intelligence algorithms, and promote the intersection and integration of related disciplines.

Gai-Ge Wang, Amir H. Alavi
Editors

mathematics

MDPI

Review

Multi-Task Optimization and Multi-Task Evolutionary Computation in the Past Five Years: A Brief Review

Qingzheng Xu [1,2], Na Wang [1], Lei Wang [3,*], Wei Li [4] and Qian Sun [4]

[1] College of Information and Communication, National University of Defense Technology, Xi'an 710106, China; xuqingzheng@hotmail.com (Q.X.); syesun@hotmail.com (N.W.)
[2] Youth Innovation Team of Shaanxi Universities, National University of Defense Technology, Xi'an 710106, China
[3] School of Mathematics and Computer Science, Shaanxi University of Technology, Hanzhong 723001, China
[4] School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China; liwei@xaut.edu.cn (W.L.); 15691751097@163.com (Q.S.)
* Correspondence: leiwang@xaut.edu.cn

Abstract: Traditional evolution algorithms tend to start the search from scratch. However, real-world problems seldom exist in isolation and humans effectively manage and execute multiple tasks at the same time. Inspired by this concept, the paradigm of multi-task evolutionary computation (MTEC) has recently emerged as an effective means of facilitating implicit or explicit knowledge transfer across optimization tasks, thereby potentially accelerating convergence and improving the quality of solutions for multi-task optimization problems. An increasing number of works have thus been proposed since 2016. The authors collect the abundant specialized literature related to this novel optimization paradigm that was published in the past five years. The quantity of papers, the nationality of authors, and the important professional publications are analyzed by a statistical method. As a survey on state-of-the-art of research on this topic, this review article covers basic concepts, theoretical foundation, basic implementation approaches of MTEC, related extension issues of MTEC, and typical application fields in science and engineering. In particular, several approaches of chromosome encoding and decoding, intro-population reproduction, inter-population reproduction, and evaluation and selection are reviewed when developing an effective MTEC algorithm. A number of open challenges to date, along with promising directions that can be undertaken to help move it forward in the future, are also discussed according to the current state. The principal purpose is to provide a comprehensive review and examination of MTEC for researchers in this community, as well as promote more practitioners working in the related fields to be involved in this fascinating territory.

Keywords: multi-task optimization; multi-task evolutionary computation; knowledge transfer; evolutionary algorithm; assortative mating; unified search space

Citation: Xu, Q.; Wang, N.; Wang, L.; Li, W.; Sun, Q. Multi-Task Optimization and Multi-Task Evolutionary Computation in the Past Five Years: A Brief Review. *Mathematics* **2021**, *9*, 864. https://doi.org/10.3390/math9080864

Academic Editor: Fabio Caraffini

Received: 22 March 2021
Accepted: 9 April 2021
Published: 14 April 2021

1. Introduction

Due to its extensive application in science and engineering fields, global optimization is a topic of great interest nowadays. Without a loss of generality, it implies the minimization of a specific objective function or fitness function [1]. Effective and common approaches for optimization problems can be mainly divided into deterministic and heuristic methods. Deterministic methods (such as linear programming and nonlinear programming) can find a global or an approximately global optimum using mathematical formulas. Generally speaking, they take advantage of the analytical properties of the optimization problem to generate a sequence of solutions that converge to a global optimum [2]. On the other hand, heuristic methods use random processes, and thus cannot guarantee the quality of the obtained solutions. Comparatively speaking, to find an acceptable solution, the deterministic approach needs fewer objective function evaluations than the stochastic approach.

However, stochastic approaches have been found to be more flexible and efficient than deterministic approaches, especially for complex "black box" problems [3].

Evolutionary algorithms (EAs) are a kind of population-based stochastic optimization methods involving the Darwinian principles of "Natural selection and survival of the fittest" [4–8]. The algorithm starts with a population of randomly generated individuals. Then, new offspring are produced iteratively by undergoing evolutionary operators such as crossover and mutation, and fitter offspring will survive to the next generation. The production and selection procedure terminates when a predefined condition is satisfied. Due to their simple implementation and strong search capability, in the last few decades, EAs have been successfully applied to solve a wide range of real-world optimization problems in areas such as defense and cybersecurity, biometrics and bioinformatics, finance and economics, sport, and games [9,10].

Despite their great successes in science and engineering, existing EAs still contain some drawbacks. One major point is that traditional EAs typically start to solve a problem from scratch, assuming a zero prior knowledge state, and focus on solving one problem at a time [11,12]. However, it is well known that real-world problems seldom exist in isolation and are usually mixed with each other. The knowledge extracted from past learning experiences can be constructively applied to solve more complex or new encountered tasks.

Traditional machine learning algorithms only work well under a common assumption that the distributions of the training and test data are the same [13]. Nevertheless, the domains, tasks, and distributions may be very different in many real-world applications. In such cases, transfer learning or multitask learning between multiple source tasks and a target task would be desirable. In contrast to tabula rasa learning, transfer learning in the field of machine learning can leverage on a pool of available data from various source tasks to improve the learning efficacy of a related target task. The fundamental motivation for transfer learning in machine learning community was discussed in a NIPS (Conference and Workshop on Neural Information Processing Systems) 1995 post-conference workshop on "Learning to Learn: Knowledge Consolidation and Transfer in Inductive Systems" [14]. Since 1995, it has attracted substantial scholar attention, and achieved significant success [13,15–17]. Although the notion of knowledge transfer or transfer learning has been prominent in machine learning, it is relatively scarce, and has received far less attention in the evolutionary computation community. Frankly speaking, a detailed description of transfer learning in machine learning is beyond the scope of this review article, which is limited in transfer learning or multi-task learning in evolutionary computation.

As a novel paradigm, transfer optimization can facilitate the automatic knowledge transfer across optimization problems [11,12]. Following from the formalization, the conceptual realizations of this paradigm are classified into three distinct categories, namely sequential transfer optimization, multi-task optimization (MTO), the main focus of this article, and multiform optimization. Note that the concept of multi-task optimization is also described using other terms such as multifactorial optimization (MFO) [18], multitasking optimization (MTO) [19], multi-task learning (MTL) [20], multitask optimization (MTO) [11], multitasking [12], evolutionary multitasking (EMT) [21], evolutionary multitasking (EMT) [22], and multifactorial operation optimization (MFOO) [23].

The basic concept of multi-task optimization was originally introduced by Prof. Ong [24]. In contrast to the traditional EAs which optimize only one task in a single run, the main idea of MTO is to solve multiple self-contained optimization tasks simultaneously. Due to its strong search capability and parallelism nature, it has attracted great research attention since it was proposed in 2015. Nevertheless, to the best of our knowledge, there is no effort being conducted on the comprehensive survey, especially in future trends and challenges, about MTO. Thus, the intention of this article is to present an attempt to fill this gap.

Up to now, no research monograph on this topic has been published, except a book chapter written by Gupta et al. [25]. The review of the literature in this paper consists of 140 articles from refereed journals and conference proceedings. These papers listed in

the bibliography are drawn from the past five years. Note that dissertations [26–29] have generally not been included, although the tendency is to be inclusive when dealing with borderline cases. One of the major concerns here is that these results and key contributions with rarely novel ideas in dissertations are usually the collection of previous results published in journals or conferences.

The remaining of this review is organized as follows. The basic definition and some confusing concepts of MTO are introduced in Section 2. In this section, we also conduct a statistical analysis of the literature. In Section 3, the mathematical analysis of conventional multi-task evolutionary computation (MTEC) is provided which theoretically explains why some existing MTECs perform better than traditional methods. Then, Section 4 describes some basic implementation approaches for MTEC, such as chromosome encoding and decoding scheme, intro-population reproduction, inter-population reproduction, balance between intra-population reproduction and inter-population reproduction, and evaluation and selection strategy. Further, related extension issues of MTEC are summarized in Section 5. In Section 6, a review of the applications of MTEC in science and engineering is conducted. Finally, the trends and challenges for further research of this exciting field are discussed in Section 7. Finally, Section 8 is devoted to main conclusions.

2. Basic Concept of Multi-Task Optimization and Multi-Task Evolutionary Computation

2.1. Definition of Multi-Task Optimization

Generally, the goal of multi-task optimization is to find the optimal solutions for multiple tasks in a single run. Without a loss of generality, suppose there are K minimization tasks to be optimized simultaneously. Specifically, denote T_i as the ith minimization task to be solved. Then, the definition of a MTO problem can be mathematically represented as follows [18]:

$$\boldsymbol{x}_i^* = \text{argmin}_{\boldsymbol{x}} T_i(\boldsymbol{x}), \quad i = 1, 2, \cdots, K \tag{1}$$

where $\boldsymbol{x}_i^*$ is a feasible solution of the ith task T_i. Note that T_i itself could be single-objective optimization or multi-objective optimization problem. A general schematic of multi-task optimization is depicted in Figure 1.

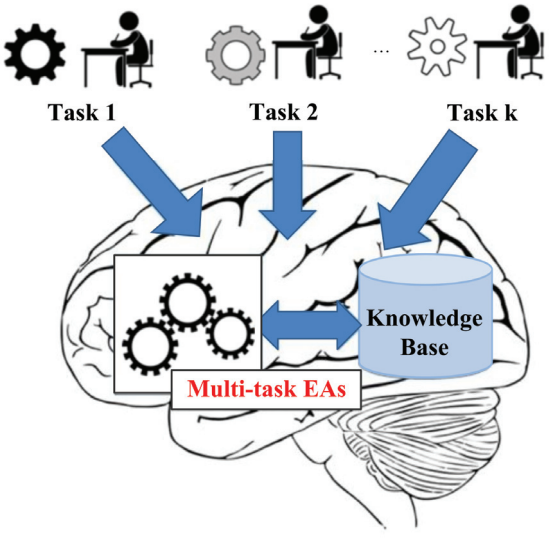


Figure 1. An illustration of a multi-task optimization problem [30].

To evaluate the individuals in MTO, several properties associated with every individual are defined as follows [18]:

Definition 1 (Factorial Cost): *The factorial cost of individual p_i on task T_j is the objective value f_j of potential solution p_i, which is denoted as ψ_j^i.*

Definition 2 (Factorial Rank): *The factorial rank of p_i on T_j is the rank index of p_i in the sorted objective value list in an ascending order, which is denoted as r_j^i.*

Definition 3 (Skill Factor): *The skill factor is defined by the index of the task assigned to an individual. The skill factor of p_i is given by $\tau_i = \text{argmin}_{j\in\{1,2,\ldots,K\}} r_j^i$.*

Definition 4 (Scalar Fitness): *The scalar fitness of p_i is the inverse of r_j^i, which is given by $\varphi_i = 1/min_{j\in\{1,2,\ldots,K\}} r_j^i$.*

Herein, the skill factor is regarded as the cultural trait which can be inherited from its parents in MTO. The scalar fitness is used as the unified performance criterion in a multi-task framework.

2.2. Confusing Concepts of MTO

As an emerging paradigm in evolutionary computation community, multi-task optimization is easy to confuse with other optimization concepts outlined and distinguished in this section.

2.2.1. Multi-Objective Optimization (MOO)

In a real-world scenario, a decision maker in the general case has to simultaneously account for multiple disparate or even contradictory criteria while selecting a particular plan of action. Mathematically, a multi-objective optimization problem can be formulated as follows:

$$\min F(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(x), \cdots, f_m(\boldsymbol{x}))^T \tag{2}$$

where $\boldsymbol{x}$ is the decision variable vector. Typically, no single optimal solution can minimize all the objectives simultaneously due to the confliction between each pair of objectives. Thus, the main purpose of an MOO problem is to obtain an optimal solution set, called a Pareto solution set, with splendid convergence and diversity.

In the literature, multi-objective evolutionary algorithms (MOEAs) that are commonly used today can be classified into three categories [31]: (a) dominance-based MOEAs, such as NSGA-II [32], (b) indicator-based MOEAs, such as HypE [33], and (c) decomposition-based MOEAs, such as MOEA/D [34].

Although MOO and MTO problems both involve the optimization of multiple objective functions, they are two distinct optimization paradigms. MOO focuses on efficiently resolving conflicts among competing objective functions in one task. As a result, solving a MOO problem typically yields a Pareto solution set that provides the best trade-offs among all objective functions. Differently, MTO aims to leverage the implicit parallelism of a population-based search to seek out the optimal solutions for two or more tasks simultaneously. Therefore, the output of a MTO problem contains two or more optimal solutions corresponding to each task.

In order to further exhibit the distinction between MOO and MTO, we refer to their population distributions in Figure 2. In real life, you can imagine a scenario where you plan to buy a cheap and fine table in a furniture store. Actually, this problem that you face is a multi-objective optimization problem. Based on the definition of Pareto optimal solution, individuals $\{p_2, p_3, p_4, p_5\}$ are incomparable to each other and are better than the individuals $\{p_1, p_6\}$ in Figure 2a. As a result, the output of this MOO problem is the Pareto

optimal solution set $\{p_2, p_3, p_4, p_5\}$, and then you can buy any table from this set based on personal preference.

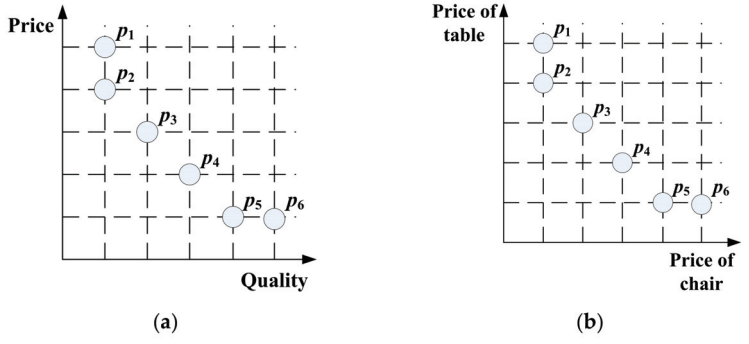


Figure 2. Population distribution for multi-objective optimization (MOO) and multi-task optimization (MTO) problems. (**a**) Multi-objective optimization problem finding a cheap and fine table. (**b**) Multi-task optimization problem finding a cheap table and a cheap chair concurrently.

In contrast, you may possibly plan to buy a cheapest table and a cheapest chair at once, which is a typical multi-task optimization problem. In Figure 2b, individuals $\{p_1, p_2\}$ are the cheapest chairs, and individuals $\{p_5, p_6\}$ are the cheapest tables in this furniture store. Thus, the output of this MTO problem is two optimal solution sets: $\{p_1, p_2\}$ and $\{p_5, p_6\}$, and then you can buy randomly ONE table from the set $\{p_5, p_6\}$ and ONE chair from the set $\{p_1, p_2\}$.

2.2.2. Sequential Transfer Optimization

The search process of many existing EAs typically begins from scratch, assuming a zero prior knowledge state. However, there is a great deal of knowledge from past exercises that can be exploited the similar search spaces in order to improve the algorithm performance. For instance, an engineering team designing a turbine for an aircraft engine would use, as a reference, past designs that have been successful and modify them accordingly to suit the current application [20].

Mathematically, we make the strict assumption that while tackling task T_K, the tasks $T_1, T_2, \ldots, T_{K-1}$ have already been addressed previously with the extracted information available in the knowledge base M [12]. Herein, T_K is said to act as the target optimization task, while $T_1, T_2, \ldots, T_{K-1}$ are said to be source tasks. As illustrated in Figure 3, the objective of sequential transfer optimization is to improve the learning of the predictive function of a target task using knowledge from any source task.

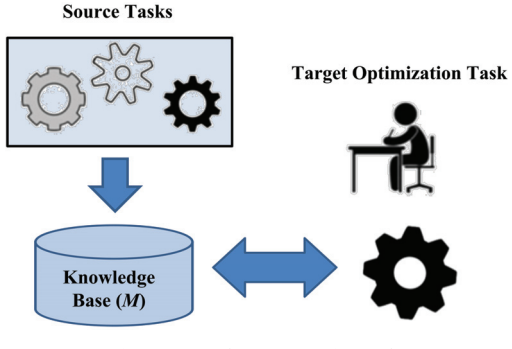


Figure 3. An illustration of a sequential transfer optimization problem [12].

2.2.3. Multi-Form Optimization

Different from multi-task optimization dealing with distinct self-contained tasks simultaneously, multi-form optimization is a novel concept for exploiting multiple alternate formulations of a single target task [12]. As illustrated in Figure 4, instead of treating each formulation independently, the basic idea of multi-form optimization is to combine different formulations into a single multi-task optimization algorithm [20].

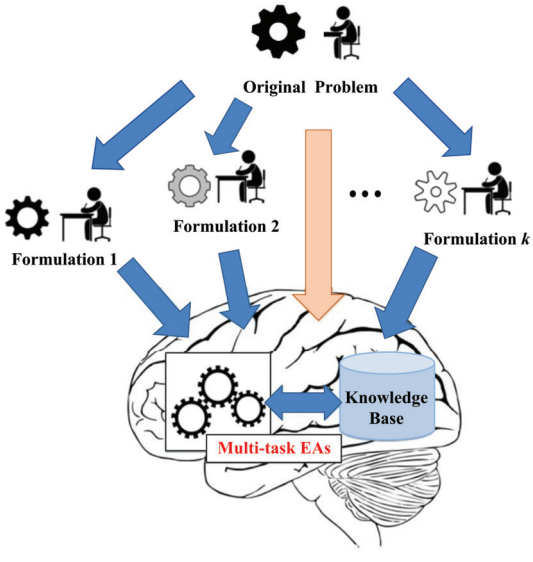


Figure 4. An illustration of multi-form optimization problem [30].

The challenge of multi-form optimization lies in the fact that it may often be difficult to ascertain which formulation is most suited for a particular problem at hand, given the known limits on computational resources. Alternate formulations induce different search behaviors, some of which may be more effective than others for a particular problem instance [30].

2.3. Multifactorial Evolutionary Algorithm

As a pioneering implementation of multi-task optimization, the multifactorial evolutionary algorithm (MFEA), inspired by the multifactorial inheritance [35,36], has gained increasing research interests due to its effectivity [18]. Algorithm 1 gives a description of the entire process of the canonical MFEA.

At the initialization phase, MFEA randomly generates a single population with $N \cdot K$ individuals in a unified search space (line 1). The individuals in the population then have a skill factor (see Definition 3 in Section 2.1), indicating the most suitable task in terms of ranking values on different tasks, and a scalar fitness (see Definition 4 in Section 2.1), determining by the reciprocal of the ranking value with respect to the most suitable task (lines 2–8).

There are two key features of MFEA, called assortative mating and selective imitation, which distinguish it from traditional EAs. The assortative mating mechanism allows not only the standard intra-task crossover between parents from the same task (lines 13–15) but also the inter-task crossover between distinct optimization instances (lines 16–18). The intensity of knowledge transfer is controlled by a user-defined parameter labeled as random mating probability (*rmp*). Since mutation is essential in genetic algorithms, MFEA with mutation applied on all newly generated candidates may achieve better performance (lines 20–23). As each newly generated individual has been assigned skill factor, the

evaluation for the individual is taken only on the task corresponded to such skill factor (line 24). After evaluation, the whole population obtain new ranking values and thus new skill factor and scalar fitness (lines 26–27), which is then used to select survivors for the next generation (line 28). Selective imitation is derived from the memetic concept of vertical cultural transmission, which aims to reduce the computational burden by evaluating an individual for their assigned task only.

Algorithm 1 Basic Structure of the Canonical MFEA

```
1   Randomly sample N·K individuals to form initial population P(0);
2   for each task T_k do
3     for every individual p_i in P(0) do
4       Evaluate p_i for task T_k;
5     end for
6   end for
7   Calculate skill factor r over population P(0);
8   Calculate scalar fitness φ according to skill factor r;
9   t = 1;
10  while stopping conditions are not satisfied do
11    while offspring generated for each task < N do
12      Sample two individuals (x_i and x_j) randomly from P(t);
13      if τ_i = τ_j then
14        [x_a, x_b] ← intra-task crossover between x_i and x_j;
15        Assign offspring x_a and x_b with skill factor τ_i(τ_j);
16      else if rand < rmp then
17        [x_a, x_b] ← inter-task crossover between x_i and x_j;
18        Assign each offspring with skill factor τ_i or τ_j randomly;
19      end if
20      [x_a] ← mutation of x_i;
21      Assign offspring x_a with skill factor τ_i;
22      [x_b] ← mutation of x_j;
23      Assign offspring x_b with skill factor τ_j;
24      Evaluate [x_a, x_b] for their assigned task only;
25    end while
26    Calculate skill factor r over population P(t);
27    Calculate scalar fitness φ according to skill factor r;
28    Select survivors to next generation;
29    t = t+1;
30  end while
```

2.4. Literature Review and Analysis

After retrieving several important full-text databases, abstract databases, and Google Scholar, 69 articles published in peer-review journals and 71 papers published in conference proceedings were collected and reviewed for this paper. The quantity of papers published each year is contained in Table 1.

Table 1. The quantity of papers published each year in the past five years. The number in parentheses represents the quantity of papers published first online.

Year	2016	2017	2018	2019	2020	Subtotal
Journal	4	4	3	20(3)	38(10)	69
Conference	12	9	12	19	19	71
Total	16	13	15	39(3)	57(10)	140

As the first paper in this field, [24] is a keynote presentation abstract published in 2016 by Springer, while the International Conference on Computational Intelligence, Cyber Security and Computational Models was held in Coimbatore, India in December 2015.

Interestingly, the first journal paper [37] was received on 1 December, 2015, and published online on 26 February, 2016, while it was published in the first volume of Complex & Intelligent Systems in 2015. For simplicity, two papers both count towards 2016, as shown in Table 1.

From Table 1, we noticed that the quantity increased for the past five years and exploded in the past two years. It had already reached 39 and 57 in 2019 and 2020, respectively, more than two thirds of the total. The results demonstrate the high research intensity and productivity in MTO, becoming a hot research topic in the evolutionary computation community.

These articles involve 277 co-authors from 12 countries, including China (184), Vietnam (19), Singapore (18), New Zealand (11), and the UK (10), as shown in Figure 5. The most prolific contributing authors in this field are summarized in Table 2. From here we see clearly that China and Singapore have demonstrated great research power in this field, and some famous research teams have emerged from China and Singapore. It is worth noting that these prominent scholars have some kind of academic connection (research scientist, Ph.D candidate, co-investigator, etc.) with the pioneer of MTO, Prof. Ong. In addition, each paper was written by 4.21 co-authors on average.

These articles were published in 34 journals and 24 international conferences. The preferential journals involve IEEE Transactions on Cybernetics (12), IEEE Transactions on Evolutionary Computation (12), IEEE Access (4), and Information Sciences (3), while the preferential conferences involve IEEE Congress on Evolutionary Computation (IEEE CEC) (33), Genetic and Evolutionary Computation Conference (GECCO) (8), and IEEE Symposium Series on Computational Intelligence (IEEE SSCI) (6). It is evident that the publication distribution shows a high concentration. The authors tend to publish these research results in the top journals and conferences in the evolution computation community, in order to promote their academic reputations. Open Access journals (like IEEE Access), meanwhile, are new options for scholars trying to seize the initiative first and achieve high visibility.

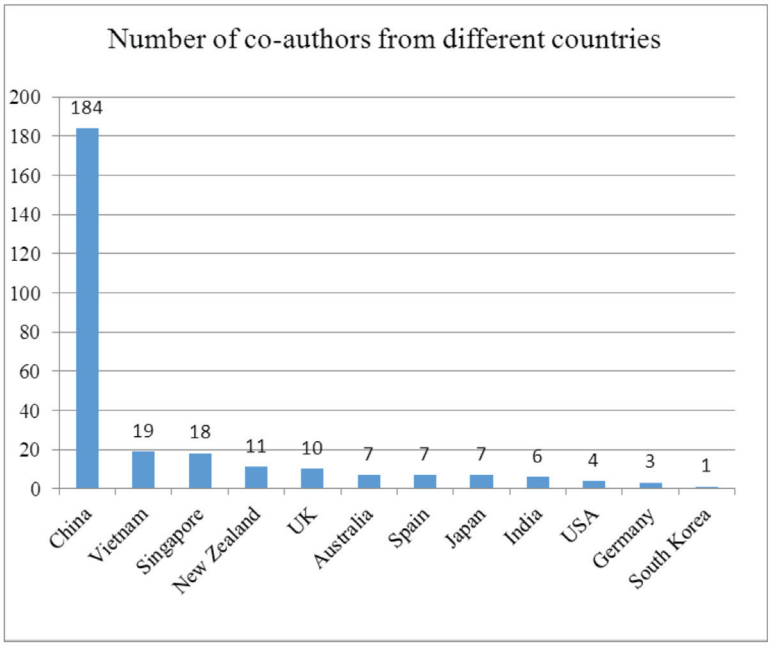


Figure 5. Number of co-authors from different countries.

Table 2. The most prolific contributing authors devoted to MTO and MTEC.

Rank	Name	Affiliations	Address	E-mail	Total Number (Journal + Conference)
1	Yew-Soon Ong	Nanyang Technological University	Singapore	asysong@ntu.edu.sg	27 (17 + 10)
2	Abhishek Gupta	Singapore Institute of Manufacturing Technology (SIMTech)	Singapore	abhishek_gupta@simtech.a-star.edu.sg	25 (17 + 8)
3	Liang Feng	Chongqing University	Chongqing, China	liangf@cqu.edu.cn	24 (13 + 11)
4	Zexuan Zhu	Shenzhen University	Shenzhen, China	zhuzx@szu.edu.cn	15 (6 + 9)
5	Jinghui Zhong	South China University of Technology	Guangzhou, China	jinghuizhong@gmail.com	13 (7 + 6)
6	Maoguo Gong	Xidian University	Xi'an, China	gong@ieee.org	11 (5 + 6)
7	Huynh Thi Thanh Binh	Hanoi University of Science and Technology	Hanoi, Vietnam	binhht@soict.hust.edu.vn	11 (4 + 7)
8	Kay Chen Tan	City University of Hong Kong	Hong Kong, China	kaytan@cityu.edu.hk	10 (7 + 3)

As of January 31, 2021, the most cited papers are [11,12,18,21,38,39], in descending order, and the other papers were cited less 70 times. Although [18] by Gupta et al. is not the first paper published in a journal or submitted to a journal, it has been widely recognized by the evolution computation community. The possible reason for this is that it provided the algorithmic background, biological foundation, basic concepts, algorithm framework, simulation experiments, and excessive experimental results of MFEA. As a result, this paper has been cited 233 times so far and considered the most classic paper in MTO and MTEC.

3. Theoretical Analyses of Multi-Task Evolutionary Computation

Experimentally, many success stories have surfaced in multi-task optimization scenarios in recent years, and demonstrated the superiority of multi-task evolutionary computation over traditional methods. A natural question is whether MTEC always improves convergence performance.

Follows directly from Holland's schema [40], under fitness proportionate selection, single-point crossover, and no mutation, the expected number of individuals in a population containing given a schema at generation is deduced in [30]. This demonstrates that, compared to conventional methods, the potential ability for MTEC to utilize knowledge transferred from other tasks in the multi-task environment to accelerate convergence towards high quality schema. Further, it was proved that the MFEA with parent-centric evolutionary operators and (μ, λ) selection can asymptotically converge to the global optimum of each constitutive task, regardless of the choice of *rmp* [41]. On the other hand, the reduction in the convergence rate of MFEA depends on the chosen *rmp* and single-task optimization may lead to faster convergence feature in the worst case.

Referring to [41], Tang et al. further proved that, by aligning two subspaces, the inter-task knowledge transfer method proposed in [42] can implicitly minimize the KL-divergence between two different subpopulations. In this way, we can implement the low-drift inter-task knowledge transfer.

In [43], adaptive model-based transfer (AMT) was proposed and analyzed theoretically. The theoretical result indicates that, by combining all available (source + target) probabilistic models, the gap between the underlying distributions of parent population and offspring population is reduced. In fact, with increasing number of source models, we can in principle make the gap arbitrarily small. Therefore, the proposed AMT framework facilitates the global convergence characteristic.

Yi et al. [44] discovered mathematically that the proposed interval dominance method has a strict transitive relation to the original method when $\gamma = 0.5$ and can be applied when comparing the dominance relationship between interval values.

The principal finding of [45] is that, for vehicle routing problems (VRPs), the positive knowledge transfer across tasks is strictly related to the intersection degree among the best

solutions. More concretely, Osaba et al. have shown that intersection degrees greater than 11% are enough for ensuring a minimum positive activity.

Recently, Lian et al. [46] provided a novel theoretical analysis and evidence of the effectiveness of MTEC. It was proved that the upper bound of expected running time for the proposed simple (4 + 2) MFEA algorithm on the $Jump_k$ function can be improved to $O(n^2 + 2^k)$ while the best upper bound for single-task optimization on the same problem is $O(n^{k-1})$. This theoretical result indicates that MTEC is probably a promising approach to deal with some distinct problems in the field of evolutionary computation. The proposed MFEA algorithm is further analyzed on several benchmark pseudo-Boolean functions [47]. Theoretical analysis results show that, by properly setting the parameter *rmp* for the group of problems with similar tasks, the upper bound of expected runtime of (4 + 2) MFEA on the harder task can be improved to be the same as on the easier one, while for the group of problems with dissimilar tasks, the expected upper bound of (4 + 2) MFEA on each task are the same as that of solving them independently. This study theoretically explains why some existing MFEAs perform better than traditional EAs.

4. Basic Implementation Approaches of Multi-Task Evolutionary Computation

Gupta and Ong [48] provided a clearer picture of the relationship between implicit genetic transfer and population diversification. The experimental results highlighted that genetic transfer is a more appropriate metaphor for explaining the success of MTEC. Da et al. [49] further considered the incorporation of gene-culture interaction to be a pivotal aspect of effective MTEC algorithms. In [50], the inheritance probability (IP) of the selective imitation was firstly defined and then the influence on MTEC algorithm was studied experimentally. To alleviate the influence of IP on the algorithm performance, an adaptive inheritance mechanism (AIM) was thus introduced to automatically adjust the IP value for different tasks at different evolutionary stages.

Solving the multi-task optimization problem in a natural way is the multipopulation evolution strategy, in which each subpopulation evolves and exploits separate search spaces independently in order to solve the corresponding task. As an example, in Figure 6, a multi-population evolution model is depicted to solve two tasks [51]. According to the multi-population evolution model of MTEC, various implementation approaches of each element proposed so far are described in detail in the following subsection.

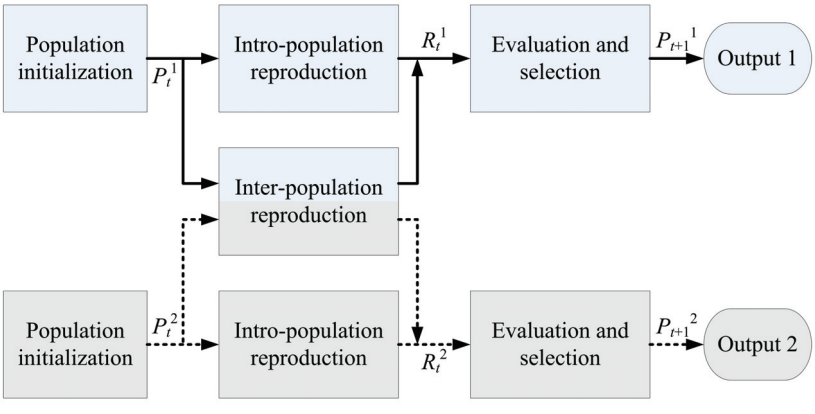


Figure 6. Multi-population evolution model for a simple case comprising two tasks [51].

4.1. Chromosome Encoding and Decoding Scheme

For effective EAs including MTEC, the unified individual representation scheme coupled with the decoding process is perhaps the most important ingredient, which directly affects the problem-solving process.

Canonical MFEA employed the unified representation scheme in a unified search space [18]. In particular, every variable of individual is simply encoded by a random key between 0 and 1 [52]. For the case of continuous optimization, decoding can be achieved in a straightforward manner by linearly mapping each random key from the genotype space to the design space of the appropriate optimization task [18,38]. For instance, consider a task T_j in which the ith variable is bounded in the range $[L_i, U_i]$. If the ith random-key of a chromosome $\boldsymbol{y}$ takes value $y_i \in [0, 1]$, then the decoding procedure is given by

$$x_i = L_i + (U_i - L_i) \cdot y_i. \tag{3}$$

In contrast, for the case of discrete optimization (such as knapsack problem (KP), quadratic assignment problem (QAP), and capacitated vehicle routing problem (CVRP)), the chromosome decoding scheme is usually problem dependent.

However, there are two obvious limitations of using a random key representation when dealing with permutation-based combinatorial optimization problems (PCOPs) [53]. Firstly, the decoding can be inefficient, since the transformation from the random key representation to the permutation is required for each fitness evaluation of EAs. Secondly, the decoding process can be highly prone to losses, since only information on relative order is derived. Therefore, Yuan et al. [53] introduced an exquisite and effective variant, called permutation based unified representation, to better adapt to PCOPs. To encode multiple VRPs, the permutation-based representation [54,55] was also adopted [56,57]. With it, a chromosome is encoded as a giant tour represented by a sequence in which each dimension is a customer id. In addition, the extended split approach [54,55] was introduced to translate a permutation-based chromosome into a feasible routing solution.

Chandra et al. [58] employed direct encoding strategy for weight representation, where all the weights are encoded in a consecutive order. Therefore, different tasks results in varied length real-parameter chromosomes in the MTEC algorithm.

The solutions offered by genetic programming (GP) are typically represented by an expression tree [59]. In the multifactorial GP (MFGP) paradigm, a novel scalable chromosome encoding scheme, gene expression representation with automatically defined functions [60], was utilized to effectively represent multiple solutions simultaneously [61]. In particular, this encoding scheme using a fixed length of strings contains one main function and multiple automatically defined functions (ADFs). The main function gives the final output, while the ADFs represent subfunctions of the main function. The corresponding decoding scheme was also proposed in [61].

Binh et al. [62] proposed an individual encoding and decoding method in unified search space for solving clustered shortest-path tree (CluSPT) problem. The number of clusters of individuals is equal to the maximum number of clusters of all tasks and the number of vertices of cluster i is the maximum number of vertexes of cluster i of all tasks. Note that such individual encoding and decoding approaches can also apply to the minimum routing cost clustered tree (CluMRCT) problem [63].

Thanh et al. [64,65] introduced the Cayley Code encoding mechanism to solve clustered tree problems. Cayley Code was chosen to be the solution representation for two reasons. The first advantage is that it can encode a solution into spanning tree easier than other methods. The other one is that it takes full advantage of existing evolutionary operators such as one-point crossover and swap-change mutation. In addition, three typical coding types in the Cayley Code families were also analyzed when performed on both single-task and multi-task optimization problems.

The Edge-sets structure has been proved to be efficient in finding spanning trees in graphs [66]. In [67], it was used to construct optimal data aggregation trees in wireless sensor networks. Each gene represents an edge, each taking a value of 0 or 1, corresponding to whether the edge is present in the spanning tree. In [68], solution presented by edge-sets representation was also built for the CluSPT problem. An individual has three properties: an ES property (edges connecting all clusters), IE property (vertices in each cluster connecting it to other vertices of different clusters), and LR property (roots of all

clusters). In order to transform a chromosome in unified search space into solutions for each task, the decoding scheme contains two separate parts. For the first task, a solution for the CluSPT problem is constructed from an individual in a unified search space by using its key properties, while the decoding method for the second task is the HBRGA algorithm proposed in [69]. However, this method cannot guarantee that the sub-graphs in clusters are also spanning trees, leading to create invalid solutions. Recently, Binh and Thanh [70] introduced another method for generating random solutions which can only produce valid solutions.

Nowadays, connectivity among communication devices in networks has been playing a significant role and multi-domain networks have been designed to help resolving scalability issues. Recently, Binh et al. [71] introduced MFEA with a new solution representation. With it, a chromosome consists of two parts in a unified search space: the first part encodes the priority of the corresponding nodes while the second part encodes the index of edges in the solution. In addition, the corresponding decoding scheme was also proposed in [71].

Constructing optimal data aggregation trees in wireless sensor networks is an NP-hard problem for larger instances. A new MFEA was proposed to solve multiple minimum energy cost aggregation tree (MECAT) problems simultaneously [67]. The authors also presented am encoding and decoding strategy, a crossover operator, and a mutation operator enabling multifactorial evolution between instances.

For solving multiple optimization tasks of fuzzy system, the encoding and decoding scheme was proposed in [72]. Each individual comprises multiple chromosomes corresponding to every fuzzy variables of the fuzzy system. Each chromosome is a series of gene sequences, and per gene has one-to-one correspondence with a membership function parameter of the fuzzy variable. When a decoding procedure is carrying out, according to the task space to be decoded, in the order that the output variable is decoded first and the input variables are decoded later, taking first few parameters of the required length from each chromosome and arranging them in ascending order, then splicing them to obtain the decoded individual.

For solving the community detection problem and active module identification problem simultaneously, a unified genetic representation and problem-specific decoding scheme was proposed [73]. An individual is encoded as an integer vector, to which each integer representing the label of community to which corresponding node is assigned.

For semantic Web service composition, a permutation-based representation was proposed [74]. A permutation is a sequence of all the services in the repository, and each service appears exactly once in the sequence. Using a forward graph building technique [75], a DAG-based solution can easily be decoded from the above permutation-based solution.

Membership function plays an important role in mining fuzzy associations. Wang and Liaw [76] proposed a structure-based representation MFEA for mining fuzzy associations. The optimization of each membership function is treated as a single task, and the proposed method can optimize all tasks in one run. More importantly, the structure based representation [77] can avoid the illegality by the transformation procedure and also reduce the number of arrangements of membership functions.

Very recently, in an evolutionary multitasking graph-based hyper-heuristic (EMHH), the chromosome of an individual is represented as a sequence of heuristics, with each bit representing a low-level heuristic [78].

4.2. Intro-Population Reproduction

As a core search operator, intro-population reproduction can significantly affect the performance of MTEC, as shown in Figure 6. The most widely utilized one is probably genetic mechanisms, namely crossover and mutation. Specifically, several typical genetic strategies include simulated binary crossover [18,79], ordered crossover (OX) [57,80], one-point crossover [59,61], DE crossover [61], guided differential evolutionary crossover [81], partially mapped crossover (PMX) and two-point crossover (TPX) [71], Gaussian mutation [18], uniform mutation [61], swap mutation (SW) [57,80], polynomial mutation [53,79],

DE mutation [61], mutation using the Powell search method [81], swap-change mutation [64], and one-point mutation [71]. The other EAs, differential evolution (DE) [82–87], particle swarm optimization (PSO) [85–94], artificial bee colony (ABC) [95], fireworks algorithm (FWA) [96], self-organized migrating algorithm (SOMA) [97], brain storm optimization (BSO) [98,99], Bat Algorithm (BA) [100], and genetic programming (GP) [61], are also utilized as fundamental algorithm for MTEC paradigms.

In addition, inspired by cooperative co-evolution genetic algorithm (CCGA), an evolutionary multi-task algorithm was proposed for the high-dimensional global optimization problem [101]. In this, a MTO problem is decomposed into multiple lower-dimensional sub-problems. In [22], the novel hyper-rectangle search strategy was designed based on the main idea of opposition-based learning. It contains two modes, which enhance the exploration ability in the unified search space and improve the exploitation ability in the sub-space of each task, respectively.

4.3. Inter-Population Reproduction

The major function of inter-population reproduction is knowledge transfer between different subpopulations, which may help to accelerate the search process and find global solutions [51]. Therefore, when, what, and how to transfer are the key issues in MTEC. An excellent MTEC algorithm should be able to deal with the three problems properly [102].

4.3.1. When to Transfer

As depicted in Figure 6, inter-population reproduction can happen at any stage of the optimization process in a multi-task scenario. Generally, the offspring are generated via genetic transfer (crossover and mutation) across tasks for each generation in [18].

In fact, knowledge transfer across tasks can also occur with a fixed generation interval along the evolution search. The interval of inter-population reproduction was set to 10 generations in EMT (evolutionary multitasking) [21], and the generation interval was fixed at 20 generations in SGDE [102]. Experimental results based on the island model revealed that better results are observed from small transfer intervals than from large transfer intervals [103].

Due to the essential differences among the landscapes of the optimization tasks, Wen and Ting [104] suggested stopping the information transfer when the parting way is detected. In MT-CPSO, if a particle within a particular population did not improve its personal best position over prescribed consecutive generations, knowledge acquired from the other task was transferred across to assist the search in more promising regions [53]. Obviously, the greater the value of the prescribed iterations is, the smaller the probability of inter-population reproduction is. Similarly, in SOMAMIF, the current optimal fitness of each population was firstly judged, and the knowledge transfer demand across tasks was triggered when the evolution process of a task stagnated for successive generations [97].

4.3.2. What to Transfer

In MFEA and its variants, each solution in every task will be selected as a transferred solution based on the same probability. The light-weight knowledge transfer strategy was proposed by Zheng et al. [105]. To be more specific, the best solutions found so far on transfer other tasks to the given task and randomly replace some individuals during the optimization process.

However, some transferred solutions, even the best solutions found so far, do not help to optimize the other tasks, thereby leading to the low efficiency of achieving the positive transfer. In evolutionary multi-task via explicit autoencoding, transferred solutions are selected from the nondominated solutions in each task [21], while the performance of this method may primarily rely on the high degree of underlying intertask similarities [41]. Recently, Lin et al. [19] proposed a new strategy for selecting valuable solutions for positive transfer. In the proposed approach, a transferred solution achieves positive transfer if it is nondominated in its target task. Then, in the original search space of this positive-transfer

solution, its several closest (based on the Euclidean distance) solutions will turn into the transferred solutions, since these solutions are more likely to achieve positive transfer.

In the existing DE-based on MTEC, the knowledge is transferred only by randomly selecting the solutions from different tasks to generate offspring without regarding the search property of DE. In fact, the successful difference vectors from the past generations can not only retain the important landscape information of the optimization problem, but also preserve the population diversity during the evolutionary process. Motivated by this consideration, Cai et al. [87] proposed a difference vector sharing mechanism for DE-based MTEC, aiming at capturing, sharing, and utilizing the knowledge of the promising difference vectors found in the evolutionary process.

More recently, Lin et al. [106] have utilized incremental Naive Bayes classifiers to select valuable solutions to be transferred during multi-task search, thus leading to the promising convergence of tasks. Furthermore, under the existing mapping strategies, tasks may be trapped in local Pareto Fronts with the guide of knowledge transfer. Thus, with the aim of improving overall convergence behavior, a randomized mapping among tasks is added that enhances the exploration capacity of transferred solutions.

Zhou et al. [107] investigated what information, except to the selective individuals, should be transferred in an MFEA framework. In particular, the difference between the individual solution and the estimated optimal solution, called the individual gradient (IG), was introduced as the additional knowledge to be transferred. The proposed approach was applied to mobile agent path planning (MAPP) [107] and the autonomous underwater vehicles (AUV) 3D path planning problem [108].

Based on a novel idea of multiproblem surrogates (MPS), an adaptive knowledge reuse framework was proposed for surrogate-assisted multi-objective optimization of computationally expensive problems [109]. The MPS provides the capability of acquiring and spontaneously transferring learned models gained from distinct but possibly related problem-solving experiences. The proposed framework consists of four primary steps: initialization, aggregation, multi-problem surrogate, and evolutionary optimization. The authors further present one possible instantiation, which utilizes a Tchebycheff aggregation approach, Gaussian process surrogate models with linear meta-regression, and an expected improvement measure to quantify the merit of evaluating a new point.

4.3.3. How to Knowledge Transfer Implicitly

As the most natural way, knowledge transfer across tasks is realized implicitly when two individuals possessing different skill factors are selected for generating the offspring via crossover. The implicit MTEC usually employs a single population with unified solution representation to solve multiple optimization tasks.

Compared with single-population SBX crossover, two parents come from two different subpopulations (P_k and P_r). Take MFEA as an example, knowledge transfer is done by inter-population SBX crossover as below [18]:

$$\boldsymbol{x}_{i*}^{k} \text{ or } \boldsymbol{x}_{i*}^{r} = \begin{cases} 0.5\Big((1+\gamma)\boldsymbol{x}_{i}^{k} + (1-\gamma)\boldsymbol{x}_{j}^{r}\Big), & rand \leq 0.5 \\ 0.5\Big((1+\gamma)\boldsymbol{x}_{j}^{r} + (1-\gamma)\boldsymbol{x}_{i}^{k}\Big), & rand > 0.5 \end{cases}. \tag{4}$$

For MT-CPSO (multitasking coevolutionary particle swarm optimization), the inter-population reproduction is provided as follows [88,92,93]:

$$\boldsymbol{x}_{i*}^{k} = 0.5\Big((1+rand)\boldsymbol{x}_{i}^{k} + (1-rand)\boldsymbol{x}_{gb}^{r}\Big) \tag{5}$$

where $\boldsymbol{x}_{i}^{k}$ and $\boldsymbol{x}_{i*}^{k}$ are the position of the i-th particle and its corresponding updated particle in subpopulation P_k, respectively, $\boldsymbol{x}_{gb}^{r}$ is the current global best position in subpopulation P_r, and $rand$ is a random number between 0 and 1.

To explore the generality of MFEA with different search mechanisms, Feng et al. [85] investigated two MTEC approaches by using PSO and DE as the search engine, respectively.

While the other genetic operators are kept the same as the original MFEA, the velocity is updated for MFPSO (multifactorial particle swarm optimization) using the following equation [85]:

$$\boldsymbol{v}_{i*}^{k} = \omega \cdot \boldsymbol{v}_{i}^{k} + c_1 \cdot rand \cdot (\boldsymbol{x}_{lb}^{k} - \boldsymbol{x}_{i}^{k}) + c_2 \cdot rand \cdot (\boldsymbol{x}_{gb}^{k} - \boldsymbol{x}_{i}^{k}) + c_3 \cdot rand \cdot (\boldsymbol{x}_{gb}^{r} - \boldsymbol{x}_{i}^{k}). \quad (6)$$

For MFDE (multifactorial differential evolution), the mutation operator with genetic materials transfer is defined as following [85]:

$$\boldsymbol{x}_{i*}^{k} = \boldsymbol{x}_{r1}^{k} + F_i(\boldsymbol{x}_{r2}^{r} - \boldsymbol{x}_{r3}^{r}). \quad (7)$$

For AMFPSO (adaptive multifactorial particle swarm optimization), the velocity is updated using the following equation [94]:

$$\boldsymbol{v}_{i*}^{k} = \omega \cdot \boldsymbol{v}_{i}^{k} + c_1 \cdot rand \cdot (\boldsymbol{x}_{lb}^{k} - \boldsymbol{x}_{i}^{k}) + c_2 \cdot rand \cdot (\boldsymbol{x}_{gb}^{k} - \boldsymbol{x}_{i}^{k}) + c_3 \cdot rand \cdot (\boldsymbol{x}_{r1}^{r} - \boldsymbol{x}_{r2}^{r}) \quad (8)$$

where $\boldsymbol{v}_{i}^{k}$ and $\boldsymbol{v}_{i*}^{k}$ are the velocity of the i-th particle and its corresponding updated particle in subpopulation P_k, respectively, $\boldsymbol{x}_{i}^{k}$ and $\boldsymbol{x}_{lb}^{k}$ are the position of the i-th particle and its best found-so-far particle in subpopulation P_k, respectively, $\boldsymbol{x}_{gb}^{k}$ is the current global best position in subpopulation P_k, $r1$ and $r2$ are random and mutually exclusive integers, c_1, c_2, c_3, and ω are four parameters to adapt to problems, and *rand* is a random number within 0 and 1.

Recently, Song et al. [90] proposed a multitasking multi-swarm optimization (MTMSO) algorithm, in which knowledge transfer across tasks was realized via arithmetic crossover on the personal best $\boldsymbol{xbest}_{i}^{k}$ of each particle among different tasks for every generation.

$$\boldsymbol{xbest}_{i*}^{k} = (1 - rand) \cdot \boldsymbol{xbest}_{i}^{k} + rand \cdot \boldsymbol{xbest}_{j}^{r} \quad (9)$$

For MPEF-SHADE (multi-population evolution framework—success-history based adaptive DE), the mutation operator with genetic materials transfer is defined as following [82,83]:

$$\boldsymbol{x}_{i*}^{k} = \boldsymbol{x}_{i}^{k} + F_i\left(\boldsymbol{x}_{gb}^{r} - \boldsymbol{x}_{i}^{k}\right) + F_i(\boldsymbol{x}_{r1}^{r} - \boldsymbol{x}_{r2}^{r}) \quad (10)$$

where $\boldsymbol{x}_{i}^{k}$ and $\boldsymbol{x}_{i*}^{k}$ are the i-th individual and the corresponding updated individual in subpopulation P_k, respectively, $\boldsymbol{x}_{gb}^{r}$ is the current best individual in subpopulation P_r, F_i is the scaling factor, and $r1$ and $r2$ are random and mutually exclusive integers.

The transfer spark was proposed to exchange information between different tasks in MTO-FWA [96]. The core idea is to bind a firework and its generated explosion sparks and guiding sparks into a task module to solve a specific problem. Based on this, assume the ith firework for the optimization task k is denoted as $\boldsymbol{FW}_{i}^{k}$ and the transfer spark generated by $\boldsymbol{FW}_{i}^{k}$ under the guiding of $\boldsymbol{TV}_{i}^{kj}$ is represented as $\boldsymbol{TS}_{i}^{kj}$. Therefore, $\boldsymbol{TV}_{i}^{kj}$ and $\boldsymbol{TS}_{i}^{kj}$ can be obtained by Equations (11) and (12), respectively

$$\boldsymbol{TV}_{i}^{kj} = \frac{2}{\sigma M_k + \sigma M_j}\left(\sum_{i=1}^{\sigma M_j} \boldsymbol{x}_{i}^{j} - \sum_{i=1}^{\sigma M_k} \boldsymbol{x}_{i}^{k}\right)\frac{r^{-\alpha}}{\sum_{r=1}^{N_k} r^{-\alpha}} \quad (11)$$

$$\boldsymbol{TS}_{i}^{kj} = \boldsymbol{FW}_{i}^{k} + \boldsymbol{TV}_{i}^{kj} \quad (12)$$

where M_k and M_j denote the total number of the individuals that the skill factor is k and j, respectively.

In order to enhance knowledge transfer among different tasks, Yin et al. [110] integrated a new cross-task knowledge transfer as following, which used a search direction from another task

$$\boldsymbol{x}_{i*}^{k} = \boldsymbol{x}_{elite}^{k} + (\boldsymbol{x}_{i}^{r} - \boldsymbol{x}_{elite}^{r}) \quad (13)$$

where x^k_{elite} and x^r_{elite} are the elite individuals of task k and r, respectively. The elite individual of the task is used to speed up the population convergence and the difference vector from another task can enhance the search diversity.

In EMT-RE framework for large-scale optimization, the knowledge transfer across tasks was conducted implicitly through the chromosomal crossover with two solutions possessing different skill factors [111]. If the current task is exactly the original task, the mutant chromosome v^p_i is simply generated from intermediate vector u_i by:

$$v^p_i = v^p_{r1} + F_i u_i \tag{14}$$

where v^p_{r1} is a randomly chosen individual from the current task, and F_i is the differential weight for controlling the amplitude of difference. If not, u_i will be mapped into the embedded space of the current task by the pseudo inverse of random embedding matrix $pinv(A_p)$:

$$v^p_i = v^p_{r1} + F_i\left(pinv(A_p)u_i\right) \tag{15}$$

where $pinv(A)$ is approximated by $\left(A^T A\right)^{-1}A^T$.

Under the existing mapping strategies, tasks may be trapped in local Pareto Fronts with the guide of the knowledge transfer. Thus, with the aim of improving overall convergence behavior, a randomized mapping among tasks was added as follows, that enhances the exploration capacity of transferred solutions [106].

$$x' = \begin{cases} \frac{(U_k - L_k)(x - L_i)}{U_i - L_i} + L_k, & r > p \\ \frac{(U_k - L_k)(x - L_i)}{U_i - L_i} + \lambda(U_k - L_k), & \text{otherwise} \end{cases} \tag{16}$$

where $\lambda \sim U[a, b]$, $r \sim U[0, 1]$, and $p \in [0, 1]$, which controls the probability of exploring the search space.

4.3.4. How to Knowledge Transfer Explicitly

In contrast to the existing implicit MTEC, the explicit MTEC algorithm employs an independent population for each optimization task and conducts knowledge transfer across tasks in an explicit manner. There are several advantages of explicit MTEC [112]. First, since each task has separate population for evolution, task-specific solution encoding schemes are employed for different tasks. Next, by only designing an explicit knowledge transfer operator, the explicit MTEC paradigm can be easily developed by employing different existing evolutionary solvers with various search capabilities for each optimization task. As different search mechanisms possess various search biases, the employment of problem-specific search operators in explicit MTEC could lead to a significantly improved algorithm performance. Further, rather than probabilistically selecting solutions for mating across tasks in the implicit MTEC, more flexible solution selection schemes, such as elite selection, can be performed before transfer in the explicit EMT for reducing negative knowledge transfer effects. However, compared with the accomplishments made in the implicit MTEC algorithms, only a few attempts have been conducted for developing the explicit MTEC approaches.

As a pioneering work, Bali et al. [113] put forward an MFEA variant with a linearized domain adaptation strategy, named LDA-MFEA, for transforming the search space of a simple task into its constitutive complex task which possesses a similar search space. The goal is to alleviate the negative transfer and to improve the quality of the generated offspring.

Feng et al. [21,114] developed an explicit MTEC algorithm to learn optimal linear mappings between different multiobjective tasks using a denoising autoencoder. In this method, different evolutionary mechanisms with different biases are cooperatively applied to solve various tasks simultaneously and the learned mappings serve as a bridge between tasks so that adaptive knowledge transfers can be conducted. By configuring the input and output layers to represent two task domains, the hidden representation provides a

possibility for conducting knowledge transfer across task domains. In particular, let $\boldsymbol{P}$ and $\boldsymbol{Q}$ represent the set of solutions uniformly and independently sampled from the search space of two different tasks T_1 and T_2, respectively. Then the mapping $\boldsymbol{M}$ from T_1 to T_2 is given by

$$\boldsymbol{M} = \left(\boldsymbol{Q}\boldsymbol{P}^T\right)\left(\boldsymbol{P}\boldsymbol{P}^T\right)^{-1}. \tag{17}$$

Therefore, the optimized solutions found for different tasks along the evolutionary search can be explicitly transferred across tasks via a simple matrix multiplication operation with the learned $\boldsymbol{M}$. The authors further improved the explicit knowledge transfer to address combinatorial optimization problems, such as VRPs [112]. In particular, they developed two mechanisms: the weighted l_1-norm-regularized learning process for capturing the transfer mapping and the solution-based knowledge transfer process across VRPs.

Aiming to strengthen the knowledge transfer efficiency, a novel genetic transform strategy was proposed and applied in individual reproduction [22]. Given two tasks T_1 and T_2, two mapping vectors $\boldsymbol{M}_{12}$ (from T_1 to T_2) and $\boldsymbol{M}_{21}$ (from T_2 to T_1) are calculated as follows:

$$\boldsymbol{M}_{21} = \left(\boldsymbol{mean}_{T_1} + \varepsilon\right)./\left(\boldsymbol{mean}_{T_2} + \varepsilon\right) \tag{18}$$

$$\boldsymbol{M}_{12} = \left(\boldsymbol{mean}_{T_2} + \varepsilon\right)./\left(\boldsymbol{mean}_{T_1} + \varepsilon\right) \tag{19}$$

where $\boldsymbol{mean}_{T_1}$ and $\boldsymbol{mean}_{T_2}$ are mean vectors of some selected individuals specific to the two tasks, respectively, and ε represents a small positive number. The operator performs element-wise division of two vectors. Based on two vectors, the parent individuals can be mapped to the vicinity of the other solutions.

It was very recently determined that a novel search space mapping mechanism, namely, subspace alignment (SA) could enable efficient and high-quality knowledge transfer among different tasks [115]. In particular, the SA strategy establishes the connection between two tasks using two transforming matrices, which can reduce the probability of negative transfer. This involves assuming there are two subpopulations $\boldsymbol{P}$ and $\boldsymbol{Q}$, with each associated with a task. They denote the source data and target data, respectively. $\boldsymbol{W}_P = \frac{1}{n}\boldsymbol{P}^T\boldsymbol{P}$ and $\boldsymbol{W}_Q = \frac{1}{n}\boldsymbol{Q}^T\boldsymbol{Q}$ denote the covariance matrices of $\boldsymbol{P}$ and $\boldsymbol{Q}$, respectively. Then $\boldsymbol{E_P}$ and $\boldsymbol{E}_Q$ consist of the set of all eigenvectors of $\boldsymbol{W_P}$ and $\boldsymbol{W}_Q$, respectively, with one eigenvector per column. From $\boldsymbol{E_P}$ and $\boldsymbol{E}_Q$, the eigenvectors corresponding to the largest h eigenvalues that can retain 95% of the information are selected to construct the subspaces of $\boldsymbol{P}$ and $\boldsymbol{Q}$, that is, $\boldsymbol{S_P}$ and $\boldsymbol{S}_Q$. Afterward, the transformation matrix $\boldsymbol{M}^*$ of mapping $\boldsymbol{S_P}$ and $\boldsymbol{S}_Q$ is obtained according to Equation (20).

$$\boldsymbol{M}^* = \boldsymbol{S_P}^T\boldsymbol{S}_Q \tag{20}$$

The transferability between two distinct tasks is effectively enhanced with a proper domain adaptation technique. However, the improper pairwise learning fashion may incur a chaotic matching problem, which dramatically degrades the inter-task mapping [110]. Keeping this in mind, a novel rank loss function for acquiring a superior inter-task mapping between the source-target instances was formulated [116]. Then, an evolutionary-path-based probabilistic representation model was proposed to represent the optimization instances. With the proposed representation model, the threat of chaotic matching between the source-target domains is effectively avoided. Finally, with a progressional Gaussian representation model, a closed-form solution of affine transformation for bridging the gap between the source-target instances was mathematically derived from the proposed rank loss function.

Recently, Chen et al. [117] proposed an evolutionary multi-task algorithm with learning task relationships (LTR) for the MOO problem. The decision space of each task is treated as a manifold, and all decision spaces of different tasks are jointly modeled as a joint manifold. The joint mapping matrix composed of multiple mapping functions is then constructed to map the decision spaces of different tasks to the latent space. Finally, the

relationships among distinct tasks can be jointly learned so as to promote the optimizing of all the tasks in a MOO problem.

Similarly, Tang et al. [42] also introduced an inter-task knowledge transfer strategy. Specifically, the low-dimension subspaces of task-specific decision spaces are first established via the principal component analysis (PCA) method. Then, the alignment matrix between two subspaces is learned and solved. After that, the corresponding solutions belonging to different tasks are projected into the subspaces. With this, two inter-task reproduction strategies are then designed in the aligned subspaces.

4.4. Balance between Intra-Population Reproduction and Inter-Population Reproduction

As illustrated in Figure 6, the offspring of individuals are generated in two ways: intra-population reproduction and inter-population reproduction. On one hand, the inductive biases transferred from another task are helpful to effectively accelerate convergence. On the other hand, excessive inter-population reproduction may lead to negative genetic transfer across tasks and bad algorithm performance [11,118]. Thus, a natural question in multi-task optimization community is finding a proper balance between intra-population reproduction and inter-population reproduction [51]. Up to now, the proposed approaches have been divided into three groups (fixed parameter, parameter adaptation, and resource reallocating) explained in the following subsections.

4.4.1. Fixed Parameter Strategy

In the original MFEA, the extent of inter-task knowledge transfer is mandated by a scalar parameter defined as the random mating probability (*rmp*), which is set as a constant of 0.3 [18]. A larger value of *rmp* induces more exploration of the entire search space, thereby facilitating population diversity. In contrast, a smaller value would encourage the exploitation of current solutions and speed up the population convergence. In TMO-MFEA, a larger *rmp* is used for diversity-related variables (DV) to enhance its diversity, while a smaller *rmp* is designed for convergence-related variables (CV) to achieve a better convergence [119,120]. Particularly, *rmp* for CV equals to 0.3, and *rmp* for DV equals to 1, which means a random assortative mating.

An appropriate parameter is essential to the efficiency and effectiveness of MTEC algorithm, and vice-versa. However, the user-defined and fixed parameter in MFEA and its variants is likely to have some distinct disadvantages. Firstly, the *rmp* parameter is manually specified based on the intuition of a decision maker. It is indeed patently clear that such an offline *rmp* assignment scheme is heavily dependent on the existence of prior knowledge about the different optimization tasks. Given the lack of prior knowledge, particularly in general black-box optimization, inappropriate (blind) *rmp* values risks the possibility of harmful inter-task knowledge transfers, thereby leading to significant performance slowdowns [41,79,121]. Secondly, the *rmp* parameter is immutably fixed for all tasks during the optimization process. Similar to biomes symbiosis [122], there are three relationships between source tasks and a target task in an MTO scenario: mutualism, parasitism, and competition. More importantly, the relationship may vary as the population distributions in their corresponding landscapes change. Although this fixed mechanism can make use of the positive knowledge transfer in some very special cases, it may intuitively bring negative effects in general cases [83].

4.4.2. Parameter Adaptation Strategy

If an optimization task is improved more times by the offspring from other tasks, the probability of knowledge transfer should be increased; otherwise, we will decrease this rate [122,123]. Thus, the probability is defined by

$$rmp_k = \frac{R_k^o}{R_k^s + R_k^o} \tag{21}$$

where R_k^s and R_k^o are the proportions of times that the current best solution in subpopulation P_k is improved by the offspring of the same task and other tasks, respectively. In addition to the transfer rate, the size of the selected candidate solutions also influences the effect of information transfer. An adaptive control mechanism for the size for each task was also devised in [123].

$$|C_k| = rmp_k(|Ofsp| - |Ofsp_k|) + |Ofsp_k| \tag{22}$$

In MPEF (multi-population evolution framework), this parameter was adaptively determined based on evolution status [82,83]:

$$rmp_k = \begin{cases} \min(rmp_k + c \cdot tsr_k,\ 1), & tsr_k > sr_k \\ \max(rmp_k - c \cdot (1 - tsr_k),\ 0), & tsr_k < sr_k \end{cases} \tag{23}$$

where sr_k is the success rate of subpopulation P_k, tsr_k is the success rate of that offspring generated with the genetic material transfer, and c is a constant parameter.

A simple random searching method was introduced to adjust this parameter [94]. The current *rmp* is stored in the candidate list when at least one of K best solutions is updated by a better solution. Otherwise, the parameter is adapted as follows:

$$rmp_k = rmp_k + \delta \cdot N(0,\ 1) \tag{24}$$

where δ is a constant parameter, and N (0,1) is a Gaussian noise with zero mean and unit variance.

Based on the saturation point of the knowledge transfer (SPKT), the knowledge transfer control scheme was proposed to control the generation of hybrid-offspring and alleviate the harmful transferred knowledge [99]. Based on the efficiencies of the global search and local search component, Liu et al. [86] proposed an adaptive control strategy, which can determine whether to perform the global search (DE) or the local search (CMA-ES) during the evolution.

Further, Binh et al. [124] proposed a new method for automatically adjusting ***rmp*** parameter. Specifically, the separate ***rmp*** value for each task is updated by

$$\boldsymbol{rmp}[i] = \frac{\boldsymbol{S}_{\tau_i, NF=0}}{NP_i} \tag{25}$$

where NP_i is number of individuals in the current task, $\boldsymbol{S}_{\tau_i, NF=0}$ is the set of individuals with skill factor τ_i and belong to the first nondominated front. The idea behind this definition is that, when most of the individuals are in the first nondominated front, the search process may get stuck in a local nondominated front and then we should increase RMP parameter for the cross-task crossover.

Besides, Zheng et al. [125] defined a novel notion of ability vector to capture the correlations between different tasks and automatically changed the intensity of knowledge transfer across tasks to enhance the performance of MTEC algorithm.

It was very recently reported that an enhanced MFEA called MFEA-II was presented, which enables an online parameter *rmp* estimation scheme in order to theoretically minimize the negative interactions between distinct optimization tasks [41]. Specifically, the extent of transfer parameter matrix is learned and adapted online based on the optimal blending of probabilistic models in a purely data-driven manner. Bali et al. [79] further presented a realization of a cognizant evolutionary multi-task engine. This framework learns inter-task relationships based on overlaps in the probabilistic search distributions derived from data generated during the search course. Recently, it was also used to solve the operation optimization of integrated energy systems [121].

Some concepts and operators of the parameter adaptation strategy utilized in MFEA-II cannot be directly applied to permutation-based discrete optimization environments, such as parent-centric interactions. Osaba et al. [126] entirely reformulated such concepts, making them suitable to deal with discrete optimization problem without losing the

inherent benefits of MFEA-II. Furthermore, dMFEA-II implements a novel and simple strategy for dynamically updating the RMP matrix to the search performance.

4.4.3. Resource Reallocating Strategy

Recently, resource reallocating strategies in MFEA were integrated, which allocate the computational resources according to the complexities of tasks. For example, Wen and Ting [104] proposed an MFEA with resource allocation, named MFEARR. It can determine the occurrence of parting ways during evolution, at which time the effective cross-task knowledge transfer begins to fail. Then, an adaption strategy was proposed, where the transformation frequency is proportional to the probability of positive knowledge transfer. Gong et al. [127] put forward a MTO-DRA algorithm to enable dynamic resources allocation according to task requirements, such that more computing resources are assigned to complex tasks. Motivated by the similar idea that the limited computing resources should be adaptively allocated to different tasks, Yao et al. [128] also proposed dynamic resource allocation strategy. During the evolution of the population, individuals with high scalar fitness will get more investments or rewards, that is, more computing resources are allocated to them, and the scalar fitness of each individual is measured by a utility and updated periodically.

4.5. Evaluation and Selection Strategy

General speaking, the complete definition of a universal selection operator is composed of evaluation, comparison, and selection methods. The individual's performance can be evaluated directly or indirectly [51]. As an indirect method, the scalar fitness was originally proposed in MFEA and its variants [18,57]. On the other hand, the fitness value of objective function is a nature and typical direct method [82,83,86,88,122]. Note that scalar fitness and function fitness are equivalence relations in a multi-task scenario [51].

After evaluating all individuals' performances (function fitness or scalar fitness), the next question is the scope or level of comparison objects. In MFEA, the *offspring-pop* (R_t) and *current-pop* (P_t) were concatenated and then a sufficient number of individuals were selected to yield a new population [18]. This approach can be called population-based (or all-to-all) comparison. As a contrast, individual-based (or one-to-one) comparison was also utilized [61,82–84,88]. Once the offspring individual is generated by intra-population or inter-population reproduction, it is compared with its parent directly and then the better one can remain in the next generation.

For the case of population-based comparison, some alternative strategies were proposed to select the fittest individuals from the joint population. For example, MFEA and its variation follow elitist selection [18], level-based selection [53], and self-adaptive parent selection [129]. Furthermore, it may remove the worse or redundant individuals so as to create more population diversity [61].

The existing MTEC algorithms adopt a fitness-based selection criterion for effectively transferring elite genes across tasks. However, population diversity is necessary when it becomes a bottleneck against the genetic transfer. In [130], Tang et al. proposed a new selection criterion keeping a balance between individual fitness and population diversity as follows:

$$\min_i\{\alpha \cdot p_i.FS + (1-\alpha) \cdot p_i.CD\} \tag{26}$$

where α is the balance factor, FS is fitness scalar which can adjust factorial cost of individuals evaluated for different tasks to a common scale, and CD is crowding distance which can approximately estimate individual diversity.

5. Related Extension Issues of Multi-Task Evolutionary Computation

5.1. Algorithm Framework

Hashimoto et al. [103] firstly explained that MFEA can be viewed as a special island model and then implemented a simple MTEC framework under the standard island model,

as illustrated in Figure 7. Note that, it is essentially an explicit multi-population structure, in which the knowledge transfer across tasks is achieved through migration periodically.

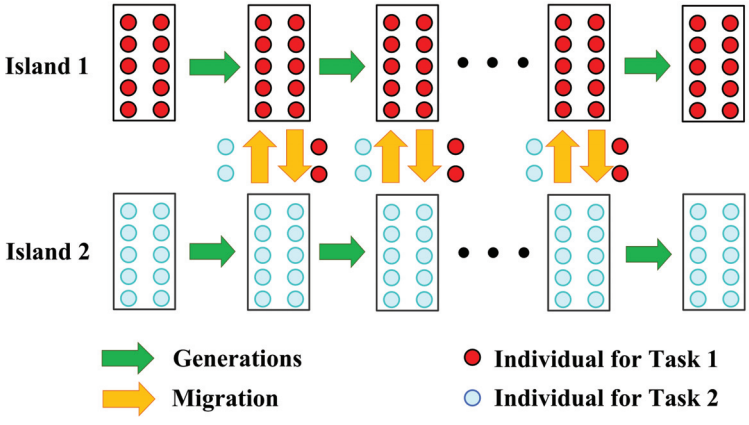


Figure 7. An illustration of the MTEC framework under the standard island model [103].

Another multi-population evolution framework (MPEF) was first established for MTO, as shown in Figure 8, wherein each population addressed its own optimization task and genetic material transfer with the other populations can be implemented and controlled in an effective manner [82,83]. Moreover, by adaptively adjusting random mating probability, it is effective for encouraging positive knowledge transfer, while avoiding negative knowledge transfer.

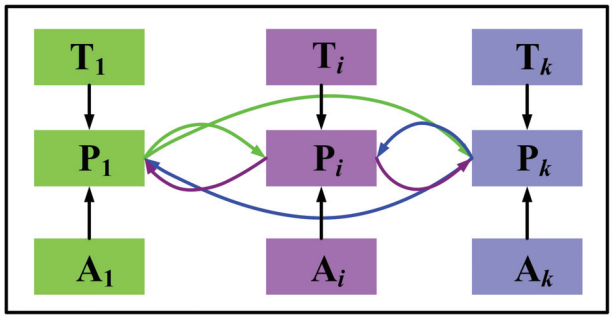


Figure 8. An illustration of the multi-population evolution framework (MPEF) [83].

Liu et al. [86] proposed an efficient surrogate-assisted multi-task memetic algorithm (SaM-MA) for solving MTO problems. In the proposed method, the population is divided into multiple sub-populations, with each sub-population focusing on solving a task. In addition, a surrogate model with the Gaussian process model is used to predict the best solution, so as to reduce the number of fitness evaluations and to improve the search efficiency.

In order to isolate the information of each task, a light-weight multi-population framework was developed, in which each population corresponds to a single task [131]. In the proposed framework depicted in Figure 9, the inter-task knowledge transfer (individual immigration) is employed to generate the offspring, and then the successful individuals (generated from the inter-task crossover and surviving in the next generation) can replace the inferior individuals of the aforementioned task.

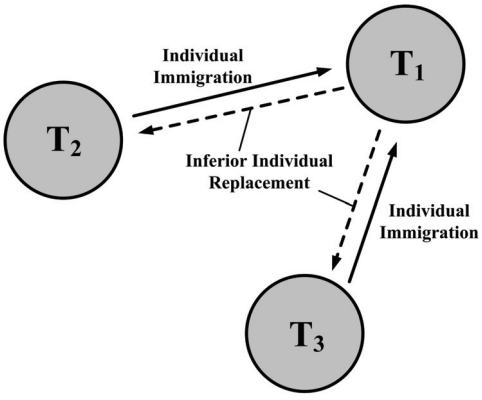


Figure 9. An illustration of the multipopulation technique for multitask optimization [131].

Besides this, research articles [84,90,100] also proposed the MTEC algorithm based on the multi-population framework, in which the number of populations is equal to the number of tasks to be optimized and each population concentrates on solving a specific task.

In order to clearly understand the focuses and differences of existing and potential works on MTEC, Jin et al. [132] proposed a general multitasking DE (MTDE) framework, which contains three major components, i.e., DE solver, knowledge transfer, and knowledge reuse. As illustrated in Figure 10, knowledge transfer is defined as both the processes of transferring knowledge out and in, and knowledge reuse as the process of utilizing the knowledge selected from the archive. In addition, two DE-specific knowledge reuse strategies were also studied in [132]: the base vector based strategy and the differential vector based strategy.

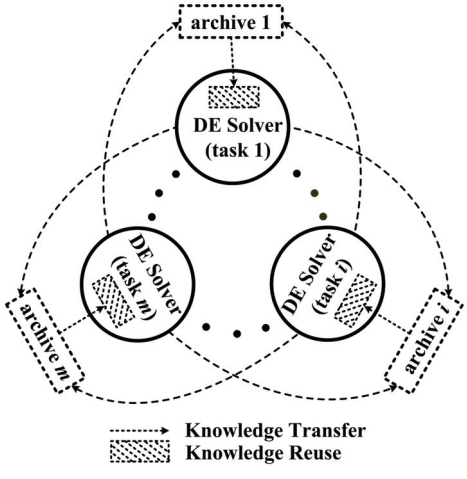


Figure 10. An illustration of multitasking DE (MTDE) framework [132].

Inspired by the cluster-based search feature of brain storm optimization (BSO), a brain storm multi-task problems solver (BSMTPS) framework was proposed by dividing individuals into several groups [99]. As illustrated in Figure 11, the offspring are generated by the internal brain storm (IBS) and the cross-task brain storm (CBS), achieving knowledge transfer within a special task and across different tasks, respectively. Zheng et al. [98] also employed the clustering technique to cluster similar solutions into one group. In this

way, it can avoid the knowledge transfer between dissimilar tasks and speed up the solving process.

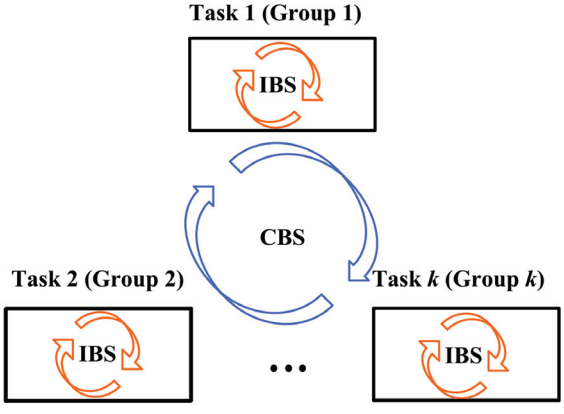


Figure 11. An illustration of the brain storm multi-task problems solver (BSMTPS) framework [99].

MFEA adopts a simple inter-task knowledge transfer with randomness and tends to suffer from excessive diversity, thereby resulting in a slow convergence speed. To deal with the above issue, a two-level transfer learning framework was proposed for MTO [133]. Particularly, the upper level performs inter-task knowledge transfer via crossover and exploited the knowledge of the elite individuals to enhance the efficiency and effectiveness of genetic transfer. The lower level is an intra-task knowledge transfer, which transmits the beneficial information from one dimension to other dimensions to improve the exploration ability of the proposed algorithm. As a result, the two levels cooperate with each other in a mutually beneficial fashion.

In order to accelerate the algorithm convergence and improve the accuracy of solutions, Xie et al. [134] introduced a hybrid algorithm combining MFEA and PSO, in which the PSO was added after genetic operation of MFEA and applied to the intermediate-pop in each generation. Furthermore, an adaptive variation adjustment factor was proposed to dynamically adjust the velocity of each particle and guarantee that the convergence velocity was not too fast.

5.2. Similarity Measure between Tasks

Some researchers have focused on analyzing and measuring task relatedness [135]. As a pioneering work in [136], the similarity between tasks for MFEA was measured from three different perspectives, i.e., the distance between best solutions, the fitness rank correlation, and the fitness landscape analysis.

Based on a correlation analysis of the objective function landscapes of distinct tasks, Gupta et al. [137] presented a synergy metric (ξ) for capturing and quantifying a promising mode of complementarity between distinct optimization tasks. The metric can explain when and why the notion of implicit genetic transfer of MTEC algorithms may lead to performance enhancements.

For classification tasks, the relatedness between tasks is estimated by comparing their most appropriate patterns [138]. Nguyen et al. [138] proposed a multiple-XOF system, which can dynamically guide the feature transfer among learning classifier systems. The proposed method improves the learning performance of individual tasks when they are related, and reduces harmful signals from other tasks when they are not supportive to a target task.

5.3. Many-Task Optimization Problem

Until now, the existing MTEC approaches mainly focused on solving two optimization tasks simultaneously and few works have been developed solving many-task optimization (MaTO) problems. The work [139] in 2016 is the first attempt to demonstrate its feasibility for solving real-world problems with more than two tasks. In an MaTO environment, a natural idea of knowledge exchange is to select the most matching individuals from all tasks [122,123]. When the number of tasks to be optimized is more than two, in order to avoid this time-consuming approach, it is important to choose the most suitable task (or assisted task) to be paired with the present task (or target task) for effective knowledge transfer. The problem of recommending an internal source task has been considered as an open challenge in a MaTO context [140].

In [102], the roulette method based on the measured similarity of each task pair was used to select the source task. In this way, one task that has high similarity with the target task has a high chance to be selected. This can reduce the harm of negative transfer because only useful knowledge is transferred.

An adaptive mechanism of choosing suitable tasks was also proposed by simultaneously considering the similarity between tasks and the accumulated rewards of knowledge transfer during evolution [141]. Based on the reliable archives storing more sufficient individuals, the similarity between different tasks is measured by the Kullback–Leibler divergence. Inspired by the idea of reinforcement learning, a reward system was further developed in the proposed framework. Finally, the most likely beneficial task is identified and transfers knowledge via a new crossover method.

As task similarity may not capture the useful knowledge between tasks, instead of using similarity measures for task selection, Shang et al. [142] proposed a task selection approach based on credit assignment to conduct positive knowledge transfer. This approach selects the appropriate task according to how good the solutions transferred from different tasks performed along the evolutionary search process. The probability of selecting task T_j to task T_i is defined by:

$$SP_j = \frac{W_{ij}}{\sum_{j=1}^{K} W_{ij}} \tag{27}$$

where an element W_{ij} gives how useful is task T_j for helping task T_i. In addition, the task assigned to individual $\boldsymbol{x}_i$ is selected by task selective probability p_i^k defined by [95]:

$$p_i^k(a) = \frac{\exp(a \cdot q_i^k)}{\sum_{k=1}^{K} \exp(a \cdot q_i^k)} \tag{28}$$

where q_i^k is the degree of how individual $\boldsymbol{x}_i$ can handle task T_k, which is defined by

$$q_i^k = \frac{N - r_i^k + 1}{\sum_{k=1}^{K} (N - r_i^k + 1)} \tag{29}$$

where r_i^k is the rank of individual $\boldsymbol{x}_i$ in task T_k.

Moreover, Tang et al. [130] proposed a group-based MFEA by clustering the similar tasks (tasks with near global optima) and dispersing the dissimilar tasks. More importantly, the genetic materials can only be transferred within the same groups so that negative genetic transfers are eliminated.

Recently, Bali et al. [79] further utilized an RMP matrix in place of a scalar parameter *rmp* to effectively many-task genetic transfers online. It offers the distinct advantage of adapting the extent of knowledge transmissions between diverse task pairs with possibly nonuniform inter-task similarities.

5.4. Decision Variable Translation Strategy

For MTO problems, the optimal solutions of all constituent tasks tend to be in different locations of the unified search space. Within the range between those optimal solutions of different tasks, the trend of those objective functions may be in different directions. As a result, the effectiveness of knowledge transfer and sharing in MTEC may degrade or even be negative in this case. The main purpose of the decision variable translation strategy is to map the optimal solution of all tasks to the center point of the unified search space so that the growth trends of all tasks are similar and facilitate knowledge transfer during the optimization process [39,143,144].

In generalized MFEA (G-MFEA), each individual in the population was translated to a new location according to Equations (30) and (31):

$$\boldsymbol{op}_i = \boldsymbol{p}_i + \boldsymbol{d}_k \tag{30}$$

$$\boldsymbol{d}_k = sf \cdot \alpha \cdot (\boldsymbol{cp} - \boldsymbol{m}_k) \tag{31}$$

where $\boldsymbol{p}_i$ and $\boldsymbol{op}_i$ ($i = 1, 2, \ldots, N_p$) are the ith solution and the corresponding transformed solution, respectively in the unified search space, N_p is the population size and the translated value $\boldsymbol{d}_k$ is estimated based on the promising solutions of the kth task. Furthermore, $\boldsymbol{m}_k$ is the estimated optimum determined by calculating the mean value of the μ percent best solutions of the kth task.

Note that the translated direction and distance are both fixed for all individuals. Unfortunately, it is easy for individuals to go beyond the legal range, and then manual efforts are required to ensure their legality. As a result, the original population distribution is destroyed inevitably. Keeping this in mind, a novel variable transformation strategy and the corresponding inverse transformation were defined as Equations (32) and (33), respectively [143,144]

$$\boldsymbol{op}_{ij} = \begin{cases} \frac{\boldsymbol{cp}_j}{\boldsymbol{m}_j} \cdot \boldsymbol{p}_{ij}, & \boldsymbol{p}_{ij} \le \boldsymbol{m}_j \\ \frac{\boldsymbol{cp}_j - 1}{\boldsymbol{m}_j - 1} \cdot \boldsymbol{p}_{ij} + \frac{\boldsymbol{m}_j - \boldsymbol{cp}_j}{\boldsymbol{m}_j - 1}, & \boldsymbol{p}_{ij} > \boldsymbol{m}_j \end{cases}, \quad j = 1, 2, \cdots, D \tag{32}$$

$$\boldsymbol{p}_{ij} = \begin{cases} \frac{\boldsymbol{m}_j}{\boldsymbol{cp}_j} \cdot \boldsymbol{op}_{ij}, & \boldsymbol{op}_{ij} \le \boldsymbol{cp}_j \\ \frac{\boldsymbol{m}_j - 1}{\boldsymbol{cp}_j - 1} \cdot \boldsymbol{op}_{ij} + \frac{\boldsymbol{cp}_j - \boldsymbol{m}_j}{\boldsymbol{cp}_j - 1}, & \boldsymbol{op}_{ij} > \boldsymbol{cp}_j \end{cases}, \quad j = 1, 2, \cdots, D \tag{33}$$

where $\boldsymbol{cp} = (0.5, 0.5, \ldots, 0.5)$ is the center point of the unified search space, $\boldsymbol{p}_i = \{\boldsymbol{p}_{i1}, \boldsymbol{p}_{i2}, \ldots, \boldsymbol{p}_{iD}\}$ is the ith solution in the original unified search space and $\boldsymbol{op}_i = \{\boldsymbol{op}_{i1}, \boldsymbol{op}_{i2}, \ldots, \boldsymbol{op}_{iD}\}$ is the corresponding ith solution in the transformed unified search space. Furthermore, $\boldsymbol{m}$ is the estimated optimal solution, which can be calculated as the mean value of the top $\mu^* N_p$ best solutions in the current generation.

5.5. Decision Variable Shuffling Strategy

In case the dimensions of decision space of different tasks in the MTO problem are different, a fine solution with small dimension may be poor and nonintegrated for task with large dimension, and some decision variables in the latter dimension of solution is always not used for tasks with small dimensions. Thus, the canonical MFEA is inefficient for MTO problems in this particular case.

To address this issue, a decision variable shuffling strategy was introduced [39]. To be specific, this strategy first randomly changes the order of the decision variables of individuals with small dimensions to give each variable an opportunity for knowledge transfer between two tasks. Then, the decision variables of individuals for the small dimensional task that are not in use are replaced with those of individuals for the large dimensional task to ensure the quality of the transferred knowledge.

Zhang and Jiang [145] systematically analyzed the defects of MFEA in dealing with heterogeneous MTO problems, and proposed the concepts of harmful transfer and defective

parents. Then hetero-dimensional assortative mating and self-adaption elite replacements were proposed to overcome these issues. On six hetero-dimensional MTO problems, the proposed algorithm performed better than other algorithms.

Generally speaking, the order of decision variables has no significant influence on the single-task EAs. In contrast, the situation is significantly different for MTEC, in which the optimization process of one task more or less influences the optimization process of other tasks. Wang et al. analyzed the influence of the order of decision variables on single-task optimization (STO) and MTO problems, respectively. In addition, three orders of decision variables were proposed in [146,147]: full reverse order, bisection reverse order, and trisection reverse order. An important feature of these orders of decision variables is that an individual can recover as himself after two times of changing the order of decision variables.

5.6. Adaptive Operator Selection Strategy

It has been found that different crossover operators have various capabilities for solving optimization problems. Therefore, the appropriate configuration of crossover is necessary for robust search performance in MFEA. Zhou et al. [148] first investigated how the different types of crossover operator used affect the knowledge transfer in MFEA on both single-objective optimization (SOO) and MOO problems. As an efficient and robust MTEC, a new MFEA with adaptive knowledge transfer (MFEA-AKT) was further proposed, in which the crossover operator employed for knowledge transfer across tasks is self-adapted based on the information collected along the evolutionary search process.

In DE, a mutant vector is obtained by perturbing a base vector with several weighted difference vectors via a certain mutation strategy. Applying different mutation operators on current population can generate different search directions and offspring populations. Multiple commonly-used mutation strategies (DE/rand/1, DE/best/1, DE/current-to-rand/1, DE/current-to-best/1, DE/rand/2, DE/best/2, and DE/best/1 + ρ) were investigated to accelerate the convergence speed in [23,115,149], where DE/best/1 + ρ is defined as follows:

$$\boldsymbol{x}_{i*}^{k} = \boldsymbol{x}_{best}^{k} + F_i(\boldsymbol{x}_{r1}^{r} - \boldsymbol{x}_{r2}^{r}) + F_i\Big(\frac{gen}{Gmax}\Big)^{a}(\boldsymbol{x}_{r3}^{r} - \boldsymbol{x}_{r4}^{r}). \tag{34}$$

In the proposed mutation strategy, the value of ρ varies from 0 to 1. Its rationale is that the current-found best solution is utilized adequately to guide the search to promising areas in the early phase, while an increased perturbation is also integrated subsequently for a diverse exploration [149]. Note that we selected the suitable mutation strategy randomly in [115] or adaptively according to their success rates in previous generations in [23].

5.7. Multi-Task Optimization under Uncertainties

Optimization problems often have different kinds of uncertainties in practice due to the influence of subjective and objective factors [150,151]. Specifically, the objective and constraint functions across tasks usually contain uncertain variables [152].

The MFEA algorithm was extended to solve the interval MTO problem under uncertainty conditions [44]. In the proposed method, an interval crowding distance based on shape evaluation is calculated to evaluate the interval solutions more comprehensively. In addition, an interval dominance relationship based on the evolutionary state is designed to obtain the interval confidence level, which considers the difference of average convergence levels and the relative size of the potential possibility between individuals.

5.8. Hyper-Heuristic Multi-Task Evolutionary Computation

Instead of searching directly in the solution space like conventional meta-heuristics, hyper-heuristics work at the higher-level search space of a set of low-level heuristics [153,154]. The goal of hyper-heuristics is to solve the problem at hand by selecting existing low-level heuristics or generating new low-level heuristics.

Although hyper-heuristics search in heuristics space, their current paradigms still focus on solving isolated optimization problems independently. To integrate the advantages of MTEC and hyper-heuristics effectively, Hao et al. [78] proposed a unified framework of the evolutionary multi-task graph-based hyper-heuristic (EMHH). Note that, in EMHH, the concept of MTEC and graph heuristics are used as the high-level search methodology and low-level heuristics, respectively. It has been evaluated on examination timetabling and graph coloring problems and the experimental results demonstrate the effectiveness and efficiency of the proposed framework.

5.9. Auxiliary Task Construction

The distinctive performance of MTEC algorithms greatly depends on the similarity of tasks in MTO problem. These methods may fail in cases where no prior knowledge on the task correlations or even no related tasks are existed. Therefore, it is worth noting that constructing the auxiliary and related task for the main task is essential to the improved performance of evolutionary search [155,156].

As the first attempt in this direction, Da et al. [80] solved a complex travel salesman problem (TSP) problem in conjunction with a closely related (but artificially generated) multi-objective optimization task in a multi-task setting. The motivation behind the proposal is that the associated MOO task can often act as a helper task which aids the search process of the original problem by leveraging upon the implicit genetic transfer. Specifically, the MOO task is formulated by decomposing the original TSP problem into two distinct sub-tours.

Similarly, vehicle routing problem with time window (VRPTW) was modeled as a two-task problem in [157], i.e., a MOO version (main task) and a single-objective version (auxiliary task). The auxiliary task provides inspiration for the creation of bone routes and semi-finished product solutions, which work together to speed up the algorithm convergence by using these illegal solutions in the search process.

Feng et al. [111] proposed an evolutionary multitasking assisted random embedding method (EMT-RE) for solving the large-scale optimization problem. Besides the original problem, several low-dimensional auxiliary tasks are constructed by random embedding to assist target optimization in a multi-task scenario.

For a given MOO problem, each single objective problem naturally shares great similarity with it [158]. Therefore, the optimization processes on these single objective functions could generate useful knowledge to enhance the problem solving process on the target MOO problem. Huang et al. [158] treated each single objective problem as a separate task domain and then discussed the detailed designs of building the dynamic domain mapping and conducting knowledge transfer from multiple single objective problems to the multi-objective problem.

In industrial production, excessive process data are generated and collected, even leading to information overload. They are predicted by models with different precision. In [119], the operational indices optimization was first established based on an accurate model (multilayer perception) and two assistant models (the first-order polynomial regression model and the second-order polynomial regression model). Note that the assistant models are alternatively used in the multi-task environment with the accurate model to realize good knowledge transfer from the assistant models to the accurate model.

Inspired by the idea of the weight function, Zheng et al. [159] introduced a new additional helper-task to accelerate the convergence of the main task in multi-task scenario. As expected, the proposed method is beneficial to positive inter-task knowledge transfer by adding possible similar tasks.

6. Applications of Multi-Task Evolutionary Computation

Since the first establishment of MFEA, a number of MTEC algorithms have been proposed and successfully applied in many benchmark problems and real-world problems over the past few years, as summarized in Table 3.

Table 3. Application domains of MTEC algorithms in the past five years.

Category	Domain	Problem	Algorithms
Benchmark problem	Continuous optimization problem	Single-objective optimization problem (SOOP)	MFEA [11], MFEA [18], None [21], MFEA-GHS [22], G-MFEA [39], MFEA-II [41], ASCMFDE [42], PGEA [49], MFDE with AIM [50], MPEF-SHADE [82], MFMP [83], MFDE [85], MFPSO [85], SaM-MA [86], MT-CPSO [86], MDE-DVSM [87], MTMSO [90], CPSOM [92], AMFPSO [94], MTO-FWA [96], MFBSO [98], BSMTO [99], BSMTO-II [99], EMTSO-CCMA [101], MFEARR [104], DEMTO [105], MFEA-DV [110], EMT-RE [111], LDA-MFEA [113], None [114], AT-MFEA [116], EBS-CMAES [122], EBSFA-CMAES [123], SREMTO [125], MTO-DRA [127], AMA [129], GMFEA [130], mMTDE [131], MTDE [132], TLTLA [133], MFEA [137], MaTDE [141], None [142], MFEA-VT [143,144], HD-MFEA [145], MFEA-FuR [146,147], MFEA-AKT [148], MFDE [149], MFEA/DE-OBL [160]
		Multiobjective optimization problem (MOOP)	EMT/ET [19], None [21], MFEA-GHS [22], AdaMOMFDE [23], MO-MFEA [38], AMTEA [43], IMFEA [44], MO-MFEA-II [79], GDE-MO-MFEA [81], MM-DE [84], MTO-FWA [96], EMTIL [106], TEMO-MPS [109], MOMFEA-SADE [115], EMT-LTR [117], TMO-MFEA [120], RPB-MO-MFEA [124], MFEA/D-DRA [128], MaTDE [141], MFEA-AKT [148], NSGAII+M [158], MO-MFEA/HELP TASK [159], MFEA/D [161], MFEA/D-M2M-SVM [162]
		Bi-level optimization problem	M-BLEA [37]
		Expensive optimization problem	MCEEA [39], MS-MTO [163]
	Discrete optimization problem	Deceptive trap function (DTF)	MF-LTGA [164]
		Clustered traveling salesman problem (CluTSP)	MF-LTGA [165]
		Vehicle routing problem (VRP)	MFEA [18], MFCGA [45], P-MFEA [56], EMA [57], EEMTA [112], dMFEA-II [126], MTO-DRA [127], MOMFMA [157]
		Quadratic assignment problem (QAP)	MFEA [11], MFEA [18], MFEA-Perm-LBS [53], MTO-DRA [127]
		Knapsack problem (KP)	MFEA [18], AMTEA [43]
		Sudoku puzzles	MFEA [48], GMFEA [130]
		Travel salesman problem (TSP)	MFEA-Perm-LBS [53], S&M-MFEA [80], COEBA [100], dMFEA-II [126]
		Linear ordering problem (LOP)	MFEA-Perm-LBS [53]
		Job-shop scheduling problem (JSP)	MFEA [11], MFEA-Perm-LBS [53], NGP [165]
		9 LOGIC suite	None [140]
		N-bit parity problem	EMTL [58]

Table 3. *Cont.*

Category	Domain	Problem	Algorithms
		Minimum routing cost clustered tree problem (CluMRCT)	MFEA [63]
		Pollution-routing problem (PRP)	None [166]
		Package delivery problem (PDP)	EEMTA [112]
		Team orienteering problem with time windows (TOPTW)	Island-EMT [167]
		Examination timetabling problem	EMHH [78]
		Graph coloring problem	EMHH [78]
		Minimum inter-cluster routing cost clustered tree problem (InterCluMRCT)	CC-MFEA [65]
		Clustered shortest path tree problem (CluSTP)	None [62], None [64], CC-MFEA [65], N-MFEA [68], N-MFEA [70]
Real-world problem	Machine learning	Time series prediction problem	MFGP [61]
		Performance prediction problem	None [168]
		Gene regulatory network (GRN) reconstruction	MMMA-FCM [169]
		Community detection	MUMI [73]
		Chaotic time series prediction problem	HD-MFEA neuroevolution [145]
		Training deep neural networks (DNN) problem	AMTO [170], None [171]
		Fuzzy cognitive map (FCM) learning	MMMA-FCM [169]
		Symbolic regression problem (SRP)	MFGP [61]
		Multi-classification problem	mXOF [138], EMC-GEP [172]
		Binary classification problem	MFGP [59]
		Automatic hyperparameter tuning of machine learning models	TEMO-MPS [109]
		Fuzzy system optimization problem	MTGFS [72]
		Association mining problem	MFEA [76]
		Classification problem	DMSPSO [89], PSO-EMT [173], MMT-ELM [174]

Table 3. *Cont.*

Category	Domain	Problem	Algorithms
	Manufacturing industry	Composites manufacturing technique	M-BLEA [37], MO-MFEA [38], MT-CPSO [88], CPSOM [92], TEMO-MPS [109]
		Pressure vessel design problem (PVDP)	MT-CPSO [88]
		Parameter extraction of photovoltaic model	SGDE [102]
		Minimum energy cost aggregation tree (MECAT) problem	ESMFA [67]
		Hyperspectral unmixing	MTSR [175], MTES [176]
		Spread spectrum radar polyphase code design (SSRPCD) problem	MFMP [83]
	Industrial engineering	Operational indices optimization of beneficiation (OIOB)	ATMO-MFEA [119]
		Continuous annealing production process (CAPL)	AdaMOMFDE [23], MFEA/D-DRA [128]
		Inter-domain path computation under domain uniqueness constraint (IDPC-DU)	MFEA [71]
		Optimal power flow (OPF) problem	MFEA [177]
		Electric power dispatch problem	MO-MFO [178]
		Well location optimization problem	AT-MFEA [116]
		Operation optimization of integrated energy system	MO-MFEA-II [121]
		Car structure design optimization problem	Multifactorial PSO-FA hybrid algorithm [91], TS+FM [95]
	Robotic	Mobile robot path planning	IMFEA [44], MFEA-IG [107,108]
		Unmanned aerial vehicle (UAV) path planning problem	MFEA [11], MO-MFEA-II [79]
	Software engineering	Search-based software test data generation (SBSTDG)	MT-EC [139]
		Cloud computing service composition (CCSC) problem	PMFEA [74], CCSC-EMA [179]
	Medicine	HIV-1 protease cleavage site prediction	None [180]
	Cybernetics	Double-pole balancing problem	MFEA-II [41], ASCMFDE [42], AMTEA [43]

6.1. Benchmark Problems

6.1.1. Continuous Optimization Problem

Evolutionary algorithms often lose their effectiveness and efficiency when applied to large-scale optimization problems. Feng et al. [111] presented a primary trial of solving large-scale optimization (up to 2000 dimensions) via the evolutionary multi-task assisted random embedding method.

EAs are not well suited for solving computationally expensive optimization problems, where the evaluation of candidate solutions needs to perform time-consuming numerical simulations or expensive physical experiments. Ding et al. [39] extended the basic MFEA to handle expensive optimization problems by transferring knowledge from multiple computationally cheap tasks to computationally expensive tasks. Similarly, a multi-surrogate based approach was adopted regarding the two surrogates as two related tasks [163]. The global surrogate model (expensive) is trained using all available data, and the local surrogate model (cheap) is trained using only part of the data subsequently selected from the data sorted.

A bi-level optimization problem (BLOP) is defined in the sense that one optimization task (the lower level problem) is nested within another (the upper level problem), which together comprise a pair of objective functions [181]. A multi-task bi-level evolutionary algorithm (M-BLEA) was provided as a promising paradigm to promote solving the upper level problem [37]. In M-BLEA, multiple lower level optimization tasks were to be appropriately solved during every generation of the upper level optimization, thereby facilitating the exploitation of underlying commonalities among them.

Although the original MFEA was designed for SOO problem [18], the idea of knowledge transfer or sharing across constitutive tasks also holds for the MOO problem. As a pioneer in multi-objective MTO, Gupta et al. [38] firstly extended the MFEA framework to the MOO domain. As a key element, a meaningful order of preference among candidate solutions in different tasks was proposed. Notice that for ordering individuals in a population, the binary preference relationship between two individuals satisfies the properties of irreflexivity, asymmetry, and transitivity [38].

Inspired by the division approach, Mo et al. [162] proposed a decomposition-based multi-objective multi-factorial evolutionary algorithm (MFEA/D-M2M). It adopts the M2M approach to decompose the MOO problem into multiple constrained sub-problems in order to enhance the population diversity. Note that a matting pool is also constructed to ensure genetic transfer across different sub-problems.

Yang et al. [120] presented the TMO-MFEA algorithm, in which decision variables were divided into two types, namely, diversity variables and convergence variables. The knowledge transfer on diversity variables is intensified to obtain evenly distributed solutions over the Pareto front (PF), whereas the knowledge transfer on convergence variables is restrained to maintain the convergence of the solution population toward the PF.

In MFEA based on decomposition strategy (MFEA/D), through multiple sets of weight vectors, each multi-objective task was decomposed into a series of SOO subtasks optimized with an independent population [161].

Recently, Ruan et al. [182] investigated when and how knowledge transfer works or fails in dynamic multi-objective optimization. Computationally knowledge transfer works poorly on problems with a fixed Pareto optimal set and under small environmental changes. In addition, the Gaussian kernel function used is not always adequate for the knowledge transfer.

6.1.2. Discrete Optimization Problem

As a preliminary attempt, several NP-hard combinatorial problems were efficiently solved within the MTEC framework, such as the traveling knapsack problem (KP) [18], Sudoku puzzles [48], travel salesman problem (TSP) [56], quadratic assignment problem (QAP) [56], linear ordering problem (LOP) [56], job-scheduling problem (JSP) [56], vehicle routing problems (VRPs) [53], and deceptive trap function (DTF) [164].

Recently, Feng et al. [57] presented a generalized variant of VRPOD, namely, the vehicle routing problem with heterogeneous capacity, time window, and occasional driver (VRPHTO), by taking the capacity heterogeneity and time window of vehicles into consideration. To illustrate its benefit, 56 new VRPHTO instances were further generated based on the existing common vehicle routing benchmarks. In addition, the stochastic team orienteering problem with time windows (TOPTW) models the trip design problem under more realistic settings by incorporating uncertainties. In [167], a new MTEC approach based on island model was developed to effectively enable knowledge sharing and transfer across search spaces.

The CluSTP problem has been solved by MFEA with new genetic operators [62,64]. In [62], the major ideas of the novel genetic operators were first constructing a spanning tree for smallest sub-graph then the spanning tree for larger sub-graph based on the spanning tree for the smaller sub-graph. Thanh et al. [64] also proposed genetic operators based on the Cayley code. Tran et al. [63] proposed a MTEC algorithm to solve multiple instances of minimum routing cost clustered tree problem (CluMRCT) together. Crossover and mutation operators were studied to create a valid solution, and a new method of calculating the CluMRCT solution was also introduced to reduce the consuming resources. More recently, Thanh et al. [68,70] further presented a novel MFEA algorithm for the CluSPT problem. Its notable feature is that the proposed MFEA has two tasks. The goal of the first task is finding the fittest solution as possible for the original problem while the goal of the second one is determining the best tree which enveloped all vertices of the problem.

Rauniyar et al. [166] put forward an MFEA based on NSGA-II to solve the pollution-routing problem (PRP). The authors considered a PRP formulation with two conflicting objectives: minimization of fuel consumption, and minimization of total travel distance.

In the literature, the n-bit parity problem is used to demonstrate the effectiveness and superiority of particular neural network architecture, training algorithms or neuroevolution methods. Chandra et al. [58] presented an evolutionary multi-task learning (EMTL) for feedforward neural networks that evolved modular network topologies for the n-bit parity problem.

6.2. Real-World Problems

6.2.1. Machine Learning

Tang et al. [174] introduced an MTEC algorithm for training multiple extreme learning machines with different number of hidden neurons for classification problem. The proposed method had achieved better quality of solutions even if some hidden neurons and connections were removed. Feature selection is an important data preprocessing technique to reduce the dimensionality in data mining and machine learning. Zhang et al. [89] proposed an ensemble classification framework based on evolutionary feature subspaces generation, which formulated the tasks of searching for the most suitable feature subspace into a MTO problem and solved it via a MTEC optimizer. Recently, MFPSO was also used to solve high-dimensional classification [173]. To be specific, two related tasks with the promising feature subset and the entire features set were developed, respectively. The MTO paradigm naturally fits the multi-classification problem by treating each binary classification problem as an optimization task within certain function evaluations. In the proposed framework, several knowledge transfer strategies (segment-based transfer, DE-based transfer, and feature transfer) were implemented to enable the interaction among the population of each separate binary task [172].

Training a deep neural network (DNN) with sophisticated architectures and a massive amount of parameters is equivalent to solving a highly complex non-convex optimization task. Zhang et al. [170] proposed a novel DNN training framework which formulated multiple related training tasks via a certain sampling method and solved them simultaneously via a MTEC algorithm. During the training process, the intermediate knowledge is identified and shared across all tasks to help their training. Recently, Martinez et al. [171]

also presented a MTEC framework to simultaneously optimize multiple deep Q learning (DQL) models.

By identifying the overlaps between communities and active modules, Chen et al. [73] revealed the complex and dynamic mechanisms of high-level biological phenomena that cannot be achieved through identifying them separately. This MTO problem contains two tasks: identification of active modules and division of network into structural communities.

The optimization problem of fuzzy systems is used to optimize the parameters or (and) structure of the fuzzy system. Zhang et al. [72] presented a general framework of the multi-task genetic fuzzy system (MTGFS) to effectively solve this problem. For the sake of better searches in multiple optimization tasks, an efficient assortative mating method (a chromosome-based shuffling strategy and a cross-task bias estimation based on shuffling) was designed according to the specialty of the membership functions.

Shen et al. [169] proposed a novel multi-objective MTEC for learning multiple large-scale fuzzy cognitive maps (FCMs) simultaneously. Each task is treated as a bi-objective problem involving both the differences between the real and learned time series and the sparsity of the whole structure.

6.2.2. Manufacturing Industry

Li et al. [175] established a multi-task sparse reconstruction (MTSR) framework to optimize multiple sparse reconstruction tasks using a single population. The proposed method aims to search the locations of nonzero components or rows instead of searching sparse vectors or matrices directly, and the intra-task and inter-task genetic transfer are employed implicitly. Besides, Zhao et al. [176] successfully handled the endmember selection of hyperspectral images.

Constructing optimal data aggregation trees in wireless sensor networks is an NP-hard problem for larger instances. A new MTEC algorithm was proposed to solve multiple minimum energy cost aggregation tree (MECAT) problems simultaneously [67]. The authors presented crossover and mutation operators, enabling multi-task evolution between instances.

6.2.3. Industrial Engineering

The operational indices optimization is crucial and difficult for the global optimization in beneficiation processes. Yang et al. [17] presented a multi-objective MFEA to solve this problem. Sampath et al. [177] also handled the optimal power flow problems with different load demands on power systems via MTEC framework. The process of continuous annealing production line is very complex in the iron and steel industry. Some environmental parameters and control variables have coupling relationships, which makes it difficult to achieve global optimization with traditional EAs. Wang and Wang [23] proposed an AdaMOMFDE algorithm based on the search mechanism of differential evolution. The optimal operation of integrated energy systems (IES) is of great significance to facilitate the penetration of distributed generators and then improve its overall efficiency. Wu et al. [121] developed a novel grid-connected IES framework by considering the biogas-solar-wind energy complementarities and solved it by MO-MFEA-II. In the Mazda multiple car design benchmark problem, three kinds of cars (SUV, CDW, and C5H) with different sizes and body shapes need to be optimized simultaneously [183]. This MTO problem was solved by two distinct MTEC algorithms [91,95].

6.2.4. Others

Thanks to the effectiveness of MTEC algorithms, they have been successfully applied to tackle other real-world problems in the literature, such as mobile robot path planning [44,107,108], search-based software test data generation [139], the cloud computing service composition problem [74,179], HIV-1 protease cleavage sites prediction [180], and the double-pole balancing problem [61–63].

7. Future Works

Although multi-task optimization methodology in the evolutionary community has been a tremendous success, compared with other well-known evolutionary and swarm intelligent methods, it is just at the stage of discipline creation and preliminary exploration in a so far unexplored research direction. Many challenges are yet to be discovered and overcome in the future in theoretical models, efficient algorithms, and engineering applications of this promising paradigm. Based on the literature analysis in the past five years, some opportunities and challenges of MTO and MTEC are summarized as follows [11,184].

7.1. Explore Mechanism of Knowledge Transfer

One of the main features of MTEC algorithms is knowledge transfer from one task to help solve other tasks, which greatly affects the optimization process and algorithm performance. Considering the general process of transfer learning, there are three key issues to be solved serially: (1) when to transfer; (2) what to transfer; (3) how to transfer.

As the original, the first question is to answer when the knowledge transfer is triggered. Theoretically, it is initiated at any stage of optimization process. Thus, the straightforward answer is executing it periodically in a fixed generation interval [21,102]. However, this trial-and-error approach does not properly explain or define the true transfer demands, leading to resource waste. Therefore, we should carefully strike a good balance between transfer cost and transfer effect. One possible and reasonable attempt in the literature is the knowledge transfer across tasks being triggered when the best solutions found so far stagnate for successive generations [88,97].

The second question might seem simple, but it is deceptively difficult. Intuitively, the best solutions found so far are good choices to be transferred. However, it might be counter-productive due to distinctly different search spaces of constitutive tasks. Inspired by biomes symbiosis, three relationships between source tasks and target tasks (mutualism, parasitism, and competition) were summarized in [83] by Li et al. Xu et al. [144] also provided a negative case when the optimal solutions were located in different positions in the unified search space. A potential approach is using the distribution characteristics of population or fitness landscape characteristics of task, instead of a special solution. These characteristics represent a full view of population or task, guiding to the global optimal solutions of each task. More importantly, the MTEC algorithm can learn these characteristics online and then adjust knowledge transfer strategy in a timely manner and properly. As a result, an important research topic is the formulation of approximate online models that can make use of the data generated during the optimization process to somehow quantify the relatedness between tasks.

The research findings of the third question are the most fruitful among three issues. In general, there are two knowledge transfer schemes in multi-task scenario in the literature: implicit transfer and explicit transfer, which are systematically discussed in Section 4.3. Although the experimental results of these schemes are encouraging, it must be kept in mind that the transfer of genetic material across tasks may be pessimistic or negative in some cases. Therefore, the mechanism of knowledge transfer across tasks should be further explored. Only by fully understanding internal mechanisms and external connections of knowledge transfer can we construct novel and positive knowledge transfer strategies.

7.2. Balance Theoretical Analysis and Practical Application

At present, most scholars concentrated mainly on algorithmic advancement and practical application. The superiority of MTEC algorithms is, in most cases, illustrated by simulation results, not by mathematical analysis with some pertinent mathematical concepts and tools. On the other hand, the researchers and practitioners ignore further study on the theoretic analysis of MTO and MTEC, either consciously or unconsciously. The most representative results focused on convergence performance [37,41] and time complexity [46,47] of simplified MFEA, which theoretically explains the superiority of the

MTEC algorithm compared with traditional single-task EAs. Comparatively speaking, other theoretical analysis (stability, diversity, etc.) of the MTEC algorithm is very limited and the distinct theoretical framework has not been assessed so far.

As a novel evolution computation paradigm, MTEC has distinct characteristics, such as a unified search space, assortative mating, and selective evaluation, to distinguish it from the single-task EAs. The intensive research of the theoretical models and functioning mechanisms of these key stone characteristics is infrequent. For this reason, the essential and fundamental development of MTO and MTEC has been hard to obtain until now.

7.3. Enhance Effectiveness and Efficiency of MTEC Algorithms

To optimize multiple tasks simultaneously, the effectiveness and adaptation of MTEC algorithm is especially important for a practitioner. In addition to canonical genetic operators (crossover, mutation, and selection), individuals encoding schemes in the unified genotype space and the implicit genetic transfer (via assortative mating and vertical cultural transmission) are the most critical ingredients of the original MFEA [18]. To improve the effectiveness and efficiency, more existing encoding schemes and genetic operators available in the literature need to be tested in a multi-task setting.

On the other hand, the performance of MTEC algorithm mainly depends on the tasks to be optimized. If the adopted methodology does not appropriately suit the behavior or feature of optimization tasks, the optimization process may be counterproductive. Therefore, we should accurately depict and deeply understand the optimization problem we face. As a critical problem to be solved urgently, based on the key feature of each task, a variety of novel encoding schemes and genetic operators can be designed to achieve the active controlling of population diversity and adaptive adjustment over the search direction of the population.

More fundamentally, we can try to modify the basic structure of the MTEC algorithm [185,186]. For instance, Chen et al. [129] introduced a local search strategy based on quasi-Newton, a re-initialization technique of worse individuals, and a self-adapt parent selection strategy to obtain better solutions. Due to the great success of memetic algorithms, incorporating local search to MTEC can also be another possible orientation. The new algorithm framework discussed in Section 5.1 can be seen as a certain positive attempt for this research topic.

7.4. Extend MTEC Algorithmic Advancements

In addition to the core demands of having suitable individuals encoding and the knowledge transfer, the advancements of peripheral elements will certainly play a crucial role in the future progress of MTO and MTEC. In this regard, some potential research prospects are in (a) the many-task optimization problem, (b) uncorrelated optimization tasks, (c) heterogeneous optimization tasks, (d) adaptively selecting the most appropriate genetic operators, (e) the multi-task optimization problem under uncertainties, (f) developing hyper-heuristic MTEC algorithms, and (g) exploring an effective approach to construct auxiliary tasks, as discussed in Section 5.

Without a doubt, these examples studied so far are just the tip of the iceberg. They are simply divided into two groups: issues similar to single-task EAs, such as (e), (f), and (g), and distinct issues in a multi-task scenario, such as (a), (b), (c), (d), and (h). Further, inspired by the single-task EAs, a good deal of similar algorithmic advancements will be explored in a multi-task scenario. For instances, adaptive MTEC is capable of adapting core mechanisms such as genetic operators, population size, and a choice of local search steps. On the other hand, several distinct forms of research in a multi-task scenario should be also conducted in the near future. For example, a natural extension of canonical MTO is effective handling of many tasks or heterogeneous tasks at a time.

7.5. Develop New Science and Engineering Applications

Finally, we believe that the notion of MTO provides a fresh perspective in terms of available knowledge transfer for improved problem solving. Several complex problems in science, engineering, operations research, etc. benefit immensely from the proposed ideas. At present, most applications focus on traditional continuous or discrete optimization fields. Thus, there is still a big gap between MTEC and the practical applications in the real world. As a preliminary attempt in the community of multi-task optimization, Prof. Ong et al. [135,187] have designed two MTO test suites for single-objective and multi-objective continuous optimization tasks, respectively. The test suite for single-objective and multi-objective MTO both contains 10 MTO complex problems, and 10 50-task MTO benchmark problems. Note that the MTO benchmark problems feature different degrees of latent synergy between their involved two component tasks.

Up to now, MTEC has not gained international recognition in community of evolutionary computation, and the reason for this might be just a lack of inspiring results in fundamental, subversive, and pioneering fields. What is more to the point, nobody has carefully and deeply considered why no breakthrough has occurred in such fields, or even summarized the basic features of MTO and MTEC.

7.6. Compare Disparate Algorithms under Different Scenarios

The No Free Lunch (NFL) theory proposed by Wolpert and Macready states that all algorithms are equivalent when their performance is evaluated over all possible problems [188]. Accordingly, each MTEC algorithm with its unique structure and operation strategy always shows different algorithm performance under different scenarios. Although some similar results have been repeatedly confirmed experimentally, it is not enough to draw a conclusion. In order to investigate the sense of the relative strengths and weaknesses of MTEC approaches, disparate strong algorithms based on a novel strategy should be compared directly and thoroughly [189].

As we all know, the overall performance of EAs more or less depends on the tested benchmark problems. Therefore, it is necessary for design diverse benchmark problems to receive a thorough investigation or evaluation. Similarly to the classical EAs, the benchmark problems for MTEC algorithms can be continuous and discrete, unimodal and multimodal, low and high dimension, static and dynamic, non-adaptive and adaptive, and with and without noise instances [152,190]. More importantly, the deviation and complementarity between any two problems should be taken into consideration. Ideally, the benchmark problems should contain various features mentioned above.

8. Conclusions

As a novel optimization paradigm proposed five years ago, with the increasing complexity and volume of data collected in the data-driven world of today, multi-task optimization appears to be an indispensable and competitive tool for the future. Since it has been proposed by Ong in 2015 [24], it has gradually attracted the attention of scholars in the community of evolutionary computation and many good results have been obtained.

To the best of our knowledge, this paper is the first literature review devoted to multi-task optimization and multi-task evolutionary computation. This overview introduced the basic definition of MTO and several confusing concepts of MTO, such as multi-objective optimization, sequential transfer optimization, and multi-form optimization. Some bold theoretical conclusions are also provided, mainly in terms of convergence performance and time complexity of some simplified forms of MFEA. Its goal is theoretically explaining the superiority of the existing MTEC algorithm compared with traditional single-task EAs.

As the core of this review article, a variety of implementation approaches of key components of MTEC are described in Section 4, including a chromosome encoding and decoding scheme, intro-population reproduction, inter-population reproduction, balance between intra-population reproduction and inter-population reproduction, and evaluation and selection strategy. In particular, we provided a clear description of inter-population

reproduction, dealing with the when, what, and how of achieving positive knowledge transfer. Further, other related extension issues of MTEC were summarized in Section 5, but they are just preliminary, fragmentary attempts and lack systematization. Next, the applications of MTEC in science and engineering were reviewed, highlighting the theoretical meaning and practical value of each problem.

Finally, a number of trends for further research and challenges that can be undertaken to help move the field forward are discussed. In a word, the future work in MTO and MTEC includes but is not limited to (1) exploring a novel mechanism of positive knowledge transfer, (2) strengthening the theoretical research to set a solid foundation, (3) enhancing the effectiveness and efficiency of MTEC algorithms by various advanced technologies, (4) extend MTEC algorithms in more complex scenarios, such as many-task or uncorrelated optimization problems under uncertainties, (5) developing real-world applications of MTEC, e.g., in machine learning, smart manufacturing [191], and smart logistics [192], and (6) comparing disparate MTEC algorithms under different scenarios.

In short, the purpose of this review article is twofold. For researchers in the evolution computation community, it provides a comprehensive review and examination of MTEC. Further, we hope to encourage more practitioners working in the related fields to become involved in this fascinating territory.

Author Contributions: Conceptualization, methodology, Q.X.; formal analysis, investigation, supervision, project administration, funding acquisition, Q.X., N.W., and L.W.; resources, data curation, W.L. and Q.S.; writing—original draft preparation, Q.X., N.W., L.W., and W.L.; writing—review and editing, Q.X. and L.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by the National Science Foundation of China 61773314, Natural Science Basic Research Program of Shaanxi 2019JZ-11 and 2020JM-709, Scientific Research Project of Education Department of Shaanxi Provincial Government 19JC011, and Research Development Foundation of Test and Training Base 23.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Molina, D.; LaTorre, A.; Herrera, F. An insight into bio-inspired and evolutionary algorithms for global optimization: Review, analysis, and lessons learnt over a decade of competitions. *Cogn. Comput.* **2018**, *10*, 517–544. [CrossRef]
2. Lin, M.-H.; Tsai, J.-F.; Yu, C.-S. A review of deterministic optimization methods in engineering and management. *Math. Probl. Eng.* **2012**, *2012*, 756023. [CrossRef]
3. Kizielewicz, B.; Sałabun, W. A new approach to identifying a multi-criteria decision model based on stochastic optimization techniques. *Symmetry* **2020**, *12*, 1551. [CrossRef]
4. Back, T.; Hammel, U.; Schwefel, H.-P. Evolutionary computation: Comments on the history and current state. *IEEE Trans. Evol. Comput.* **1997**, *1*, 3–17. [CrossRef]
5. Jin, Y.C.; Branke, J. Evolutionary optimization in uncertain environments—A survey. *IEEE Trans. Evol. Comput.* **2005**, *9*, 303–317. [CrossRef]
6. Nguyen, T.T.; Yang, S.X.; Branke, J. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm Evol. Comput.* **2012**, *6*, 1–24. [CrossRef]
7. Tanabe, R.; Ishibuchi, H. A review of evolutionary multimodal multiobjective optimization. *IEEE Trans. Evol. Comput.* **2020**, *24*, 193–200. [CrossRef]
8. Li, J.; Lei, H.; Alavi, A.H.; Wang, G.-G. Elephant herding optimization: Variants, hybrids, and applications. *Mathematics* **2020**, *8*, 1415. [CrossRef]
9. Bennis, F.; Bhattacharjya, R.K. *Nature-Inspired Methods for Metaheuristics Optimization: Algorithms and Applications in Science and Engineering*; Springer Nature: Basingstoke, UK, 2020.
10. Mirjalili, S.; Dong, J.S.; Lewis, A. *Nature-Inspired Optimizers: Theories, Literature Reviews and Applications*; Springer Nature: Basingstoke, UK, 2020.
11. Ong, Y.-S.; Gupta, A. Evolutionary multitasking: A computer science view of cognitive multitasking. *Cogn. Comput.* **2016**, *8*, 125–142. [CrossRef]

12. Gupta, A.; Ong, Y.-S.; Feng, L. Insights on transfer optimization: Because experience is the best teacher. *IEEE Trans. Emerg. Top. Comput. Intell.* **2018**, *2*, 51–64. [CrossRef]
13. Pan, S.J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359. [CrossRef]
14. NIPS*95 Post-Conference Workshop. Available online: http://socrates.acadiau.ca/courses/comp/dsilver/NIPS95_LTL/transfer.workshop.1995.html (accessed on 31 March 2021).
15. Caruana, R. Multitask learning. In *Learning to Learn*; Thrun, S., Pratt, L., Eds.; Springer: New York, NY, USA, 1998; pp. 95–133.
16. Weiss, K.; Khoshgoftaar, T.M.; Wang, D.D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 9. [CrossRef]
17. Thrun, S. Is learning the n-th thing any easier than learning the first. In *Advances in Neural Information Processing Systems*; Mozer, M.C., Jordan, M.I., Petsche, T., Eds.; The MIT Press: Cambridge, MA, USA, 1996; pp. 640–646.
18. Gupta, A.; Ong, Y.-S.; Feng, L. Multifactorial evolution: Toward evolutionary multitasking. *IEEE Trans. Evol. Comput.* **2016**, *20*, 343–357. [CrossRef]
19. Lin, J.B.; Liu, H.L.; Tan, K.C.; Gu, F.Q. An effective knowledge transfer approach for multiobjective multitasking optimization. *IEEE Trans. Cybern.* **2020**, in press. Available online: https://ieeexplore.ieee.org/document/9032363/ (accessed on 11 March 2020). [CrossRef]
20. Min, A.T.W.; Sagarna, R.; Gupta, A.; Ong, Y.-S.; Goh, C.K. Knowledge transfer through machine learning in aircraft design. *IEEE Comput. Intell. Mag.* **2017**, *12*, 48–60. [CrossRef]
21. Feng, L.; Zhou, L.; Zhong, J.H.; Gupta, A.; Ong, Y.-S.; Tan, K.C.; Qin, A.K. Evolutionary multitasking via explicit autoencoding. *IEEE Trans. Cybern.* **2019**, *49*, 3457–3470. [CrossRef]
22. Liang, Z.P.; Zhang, J.; Feng, L.; Zhu, Z.X. A hybrid of genetic transform and hyper-rectangle search strategies for evolutionary multi-tasking. *Expert Syst. Appl.* **2019**, *138*, 1–18. [CrossRef]
23. Wang, Z.; Wang, X.P. Multiobjective multifactorial operation optimization for continuous annealing production process. *Ind. Eng. Chem. Res.* **2019**, *58*, 19166–19178. [CrossRef]
24. Ong, Y.-S. Towards evolutionary multitasking: A new paradigm in evolutionary computation. In Proceedings of the International Conference on Computational Intelligence, Cyber Security and Computational Models, Coimbatore, India, 17–19 December 2015; pp. 25–26.
25. Gupta, A.; Da, B.S.; Yuan, Y.; Ong, Y.-S. On the emerging notion of evolutionary multitasking: A computational analog of cognitive multitasking. In *Recent Advances in Evolutionary Multi-Objective Optimization*; Bechikh, S., Datta, R., Gupta, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2017; pp. 139–157.
26. Cheng, M.Y. Attribute Selection Method Based on Binary Ant Colony Optimization and Fractal Dimension. Ph.D. Thesis, Hefei University of Technology, Hefei, China, 2017. (In Chinese).
27. Chen, W.Q. Active Module Identification in Biological Networks. Ph.D. Thesis, University of Birmingham, Birmingham, UK, 2018.
28. Min, A.T.W. Transfer Optimization in Complex Engineering Design. Ph.D. Thesis, Nanyang Technological University, Singapore, 2019.
29. Da, B.S. Methods in Multi-Source Data-Driven Transfer Optimization. Ph.D. Thesis, Nanyang Technological University, Singapore, 2019.
30. Gupta, A.; Ong, Y.-S. Back to the roots: Multi-x evolutionary computation. *Cogn. Comput.* **2019**, *11*, 1–17. [CrossRef]
31. Trivedi, A.; Srinivasan, D.; Sanyal, K.; Ghosh, A. A survey of multiobjective evolutionary algorithms based on decomposition. *IEEE Trans. Evol. Comput.* **2017**, *21*, 440–462. [CrossRef]
32. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
33. Bader, J.; Zitzler, E. HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evol. Comput.* **2011**, *19*, 45–76. [CrossRef] [PubMed]
34. Zhang, Q.F.; Li, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **2007**, *11*, 712–731. [CrossRef]
35. Rice, J.; Cloninger, C.R.; Reich, T. Multifactorial inheritance with cultural transmission and assortative mating. I. Description and basic properties of the unitary models. *Am. J. Hum. Genet.* **1978**, *30*, 618–643.
36. Cloninger, C.R.; Rice, J.; Reich, T. Multifactorial inheritance with cultural transmission and assortative mating. II. A general model of combined polygenic and cultural inheritance. *Am. J. Hum. Genet.* **1979**, *31*, 176–198.
37. Gupta, A.; Mańdziuk, J.; Ong, Y.-S. Evolutionary multitasking in bi-level optimization. *Complex Intell. Syst.* **2015**, *1*, 83–95. [CrossRef]
38. Gupta, A.; Ong, Y.-S.; Feng, L.; Tan, K.C. Multiobjective multifactorial optimization in evolutionary multitasking. *IEEE Trans. Cybern.* **2017**, *47*, 1652–1665. [CrossRef]
39. Ding, J.L.; Yang, C.E.; Jin, Y.C.; Chai, T.Y. Generalized multi-tasking for evolutionary optimization of expensive problems. *IEEE Trans. Evol. Comput.* **2019**, *23*, 44–58. [CrossRef]
40. Bridges, C.L.; Goldberg, D.E. An analysis of reproduction and crossover in a binary-coded genetic algorithm. In Proceedings of the International Conference on Genetic Algorithms and Their Application, Cambridge, MA, USA, 28–31 July 1987; pp. 9–13.
41. Bali, K.K.; Ong, Y.-S.; Gupta, A.; Tan, P.S. Multifactorial Evolutionary Algorithm with Online Transfer Parameter Estimation: MFEA-II. *IEEE Trans. Evol. Comput.* **2020**, *24*, 69–83. [CrossRef]

42. Tang, Z.D.; Gong, M.G.; Wu, Y.; Liu, W.F.; Xie, Y. Regularized evolutionary multitask optimization: Learning to intertask transfer in aligned subspace. *IEEE Trans. Evol. Comput.* **2020**, *25*, 262–276. [CrossRef]
43. Da, B.S.; Gupta, A.; Ong, Y.-S. Curbing negative influences online for seamless transfer evolutionary optimization. *IEEE Trans. Cybern.* **2019**, *49*, 4365–4378. [CrossRef]
44. Yi, J.; Bai, J.R.; He, H.B.; Zhou, W.; Yao, L.Z. A multifactorial evolutionary algorithm for multitasking under interval uncertainties. *IEEE Trans. Evol. Comput.* **2020**, *24*, 908–922. [CrossRef]
45. Osaba, E.; Martinez, A.D.; Lobo, J.L.; Lana, I.; Ser, J.D. On the transferability of knowledge among vehicle routing problems by using cellular evolutionary multitasking. In Proceedings of the IEEE International Conference on Intelligent Transportation Systems, Rhodes, Greece, 20–23 September 2020; pp. 1–8.
46. Lian, Y.C.; Huang, Z.X.; Zhou, Y.R.; Chen, Z.F. Improve theoretical upper bound of Jumpk function by evolutionary multitasking. In Proceedings of the High Performance Computing and Cluster Technologies Conference, Guangzhou, China, 22–24 June 2019; pp. 44–50.
47. Huang, Z.X.; Chen, Z.F.; Zhou, Y.R. Analysis on the efficiency of multifactorial evolutionary algorithms. In Proceedings of the International Conference on Parallel Problem Solving from Nature, Glasgow, UK, 19–24 July 2020; pp. 634–647.
48. Gupta, A.; Ong, Y.-S. Genetic transfer or population diversification? Deciphering the secret ingredients of evolutionary multitask optimization. In Proceedings of the IEEE Symposium Series on Computational Intelligence, Athens, Greece, 6–9 December 2016; pp. 1–7.
49. Da, B.S.; Gupta, A.; Ong, Y.-S.; Feng, L. The boon of gene-culture interaction for effective evolutionary multitasking. In Proceedings of the Australasian Conference on Artificial Life and Computational Intelligence, Canberra, Australia, 2–5 February 2016; pp. 54–65.
50. Peng, D.M.; Cai, Y.Q.; Fu, S.K.; Luo, W. Experimental analysis of selective imitation for multifactorial differential evolution. In Proceedings of the International Conference on Bio-Inspired Computing: Theories and Applications, Zhengzhou, China, 22–25 November 2019; pp. 15–26.
51. Wang, N.; Xu, Q.Z.; Fei, R.; Yang, J.G.; Wang, L. Rigorous analysis of multi-factorial evolutionary algorithm as multi-population evolution model. *Int. J. Comput. Intell. Syst.* **2019**, *12*, 1121–1133. [CrossRef]
52. Bean, J.C. Genetic algorithms and random keys for sequencing and optimization. *Orsa J. Comput.* **1994**, *6*, 154–160. [CrossRef]
53. Yuan, Y.; Ong, Y.-S.; Gupta, A.; Tan, P.S.; Xu, H. Evolutionary multitasking in permutation-based combinatorial optimization problems: Realization with TSP, QAP, LOP, and JSP. In Proceedings of the IEEE Region 10 Conference, Singapore, 22–25 November 2016; pp. 3157–3164.
54. Mirabi, M. A novel hybrid genetic algorithm for the multidepot periodic vehicle routing problem. *Artif. Intell. Eng. Des. Anal. Manuf. Aiedam* **2014**, *29*, 45–54. [CrossRef]
55. Prins, C. Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Eng. Appl. Artif. Intell.* **2009**, *22*, 916–928. [CrossRef]
56. Zhou, L.; Feng, L.; Zhong, J.H.; Ong, Y.-S.; Zhu, Z.X.; Sha, E. Evolutionary multitasking in combinatorial search spaces: A case study in capacitated vehicle routing problem. In Proceedings of the IEEE Symposium Series on Computational Intelligence, Athens, Greece, 6–9 December 2016; pp. 1–8.
57. Feng, L.; Zhou, L.; Gupta, A.; Zhong, J.H.; Zhu, Z.X.; Tan, K.C.; Qin, K. Solving generalized vehicle routing problem with occasional drivers via evolutionary multitasking. *IEEE Trans. Cybern.* **2019**, in press. Available online: https://ieeexplore.ieee.org/document/8938734 (accessed on 23 December 2019). [CrossRef]
58. Chandra, R.; Gupta, A.; Ong, Y.-S.; Goh, C.K. Evolutionary multi-task learning for modular training of feedforward neural networks. In Proceedings of the International Conference on Neural Information Processing, Kyoto, Japan, 16–21 October 2016; pp. 37–46.
59. Wen, Y.-W.; Ting, C.-K. Learning ensemble of decision trees through multifactorial genetic programming. In Proceedings of the IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, 24–29 July 2016; pp. 5293–5300.
60. Zhong, J.H.; Ong, Y.-S.; Cai, W.T. Self-learning gene expression programming. *IEEE Trans. Evol. Comput.* **2016**, *20*, 65–80. [CrossRef]
61. Zhong, J.H.; Feng, L.; Cai, W.T.; Ong, Y.-S. Multifactorial genetic programming for symbolic regression problems. *IEEE Trans. Syst. ManCybern. Syst.* **2020**, *50*, 4492–4505. [CrossRef]
62. Binh, H.T.T.; Thanh, P.D.; Trung, T.B.; Thao, L.P. Effective multifactorial evolutionary algorithm for solving the cluster shortest path tree problem. In Proceedings of the IEEE Congress on Evolutionary Computation, Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
63. Trung, T.B.; Thanh, L.T.; Hieu, L.T.; Thanh, P.D.; Binh, H.T.T. Multifactorial evolutionary algorithm for clustered minimum routing cost problem. In Proceedings of the International Symposium on Information and Communication Technology, Hanoi, Vietnam, 4–6 December 2019; pp. 170–177.
64. Thanh, P.D.; Dung, D.A.; Tien, T.N.; Binh, H.T.T. An effective representation scheme in multifactorial evolutionary algorithm for solving cluster shortest-path tree problem. In Proceedings of the IEEE Congress on Evolutionary Computation, Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
65. Thanh, P.D.; Binh, H.T.T.; Trung, T.B.; Long, N.B. Multifactorial evolutionary algorithm for solving clustered tree problems: Competition among Cayley codes. *Memetic Comput.* **2020**, *12*, 185–217.

66. Raidl, G.R.; Julstrom, B.A. Edge sets: An effective evolutionary coding of spanning trees. *IEEE Trans. Evol. Comput.* **2003**, *7*, 225–239. [CrossRef]
67. Tam, N.T.; Tuan, T.Q.; Binh, H.T.T.; Swami, A. Multifactorial evolutionary optimization for maximizing data aggregation tree lifetime in wireless sensor networks. In Proceedings of the Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II, Online Only, CA, USA, 27 April–9 May 2020; pp. 114130Z:1–114130Z:14.
68. Thanh, P.D.; Binh, H.T.T.; Trung, T.B. An efficient strategy for usingmultifactorial optimization to solve the clustered shortest path tree problem. *Appl. Intelliigence* **2020**, *50*, 1233–1258. [CrossRef]
69. Binh, H.T.T.; Thanh, P.D.; Thang, T.B. New approach to solving the clustered shortest-path tree problem based on reducing the search space of evolutionary algorithm. *Knowl. -Based Syst.* **2019**, *180*, 12–25. [CrossRef]
70. Binh, H.T.T.; Thanh, P.D. Two levels approach based on multifactorial optimization to solve the clustered shortest path tree problem. *Evol. Intell.* **2020**, in press. Available online: https://link.springer.com/article/10.1007/s12065-020-00501-w (accessed on 14 October 2020).
71. Binh, H.T.T.; Thang, T.B.; Long, N.B.; Hoang, N.V.; Thanh, P.D. Multifactorial evolutionary algorithm for inter-domain path computation under domain uniqueness constraint. In Proceedings of the IEEE Congress on Evolutionary Computation, Glasgow, UK, 19–24 July 2020; pp. 1–8.
72. Zhang, K.; Hao, W.N.; Yu, X.H.; Jin, D.W.; Zhang, Z.H. A multitasking genetic algorithm for mamdani fuzzy system with fully overlapping triangle membership functions. *Int. J. Fuzzy Syst.* **2020**, *22*, 2449–2465. [CrossRef]
73. Chen, W.Q.; Zhu, Z.X.; He, S. MUMI: Multitask module identification for biological networks. *IEEE Trans. Evol. Comput.* **2020**, *24*, 765–776. [CrossRef]
74. Wang, C.; Ma, H.; Chen, G.; Hartmann, S. Evolutionary multitasking for semantic web service composition. In Proceedings of the IEEE Congress on Evolutionary Computation, Wellington, New Zealand, 10–13 June 2019; pp. 2490–2497.
75. Wang, C.; Ma, H.; Chen, A.; Hartmann, S. Comprehensive quality-aware automated semantic web service composition. In Proceedings of the Australasian Joint Conference on Artificial Intelligence, Melbourne, Australia, 19–20 August 2017; pp. 195–207.
76. Wang, T.-C.; Liaw, R.-T. Multifactorial genetic fuzzy data mining for building membership functions. In Proceedings of the IEEE Congress on Evolutionary Computation, Glasgow, UK, 19–24 July 2020; pp. 1–8.
77. Ting, C.-K.; Wang, T.-C.; Liaw, R.-T.; Hong, T.-P. Genetic algorithm with a structure-based representation for genetic-fuzzy data mining. *Soft Comput.* **2017**, *21*, 2871–2882. [CrossRef]
78. Hao, X.X.; Qu, R.; Liu, J. A unified framework of graph-based evolutionary multitasking hyper-heuristic. *IEEE Trans. Evol. Comput.* **2021**, *25*, 35–47. [CrossRef]
79. Bali, K.K.; Gupta, A.; Ong, Y.-S.; Tan, P.S. Cognizant multitasking in multiobjective multifactorial evolution: MO-MFEA-II. *IEEE Trans. Cybern.* **2020**, *51*, 1784–1796. [CrossRef]
80. Da, B.S.; Gupta, A.; Ong, Y.-S.; Feng, L. Evolutionary multitasking across single and multi-objective formulations for improved problem solving. In Proceedings of the IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, 24–29 July 2016; pp. 1695–1701.
81. Tuan, N.Q.; Hoang, T.D.; Binh, H.T.T. A guided differential evolutionary multi-tasking with powell search method for solving multi-objective continuous optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
82. Li, G.H.; Zhang, Q.F.; Gao, W.F. Multipopulation evolution framework for multifactorial optimization. In Proceedings of the Genetic and Evolutionary Computation Conference, Kyoto, Japan, 15–19 July 2018; pp. 215–216.
83. Li, G.H.; Lin, Q.Z.; Gao, W.F. Multifactorial optimization via explicit multipopulation evolutionary framework. *Inf. Sci.* **2020**, *512*, 1555–1570. [CrossRef]
84. Chen, Y.L.; Zhong, J.H.; Tan, M.K. A fast memetic multi-objective differential evolution for multi-tasking optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
85. Feng, L.; Zhou, W.; Zhou, L.; Jiang, S.W.; Zhong, J.H.; Da, B.S.; Zhu, Z.X.; Wang, Y. An empirical study of multifactorial PSO and multifactorial DE. In Proceedings of the IEEE Congress on Evolutionary Computation, San Sebastian, Spain, 5–8 June 2017; pp. 921–928.
86. Liu, D.N.; Huang, S.J.; Zhong, J.H. Surrogate-assisted multi-tasking memetic algorithm. In Proceedings of the IEEE Congress on Evolutionary Computation, Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
87. Cai, Y.Q.; Peng, D.M.; Fu, S.K.; Tian, H. Multitasking differential evolution with difference vector sharing mechanism. In Proceedings of the IEEE Symposium Series on Computational Intelligence, Xiamen, China, 6–9 December 2019; pp. 3309–3346.
88. Cheng, M.Y.; Gupta, A.; Ong, Y.-S.; Ni, Z.W. Coevolutionary multitasking for concurrent global optimization: With case studies in complex engineering design. *Eng. Appl. Artif. Intell.* **2017**, *64*, 13–24. [CrossRef]
89. Zhang, B.Y.; Qin, A.K.; Sellis, T. Evolutionary feature subspaces generation for ensemble classification. In Proceedings of the Genetic and Evolutionary Computation Conference, Kyoto, Japan, 15–19 July 2018; pp. 577–584.
90. Song, H.; Qin, A.K.; Tsai, P.-W.; Liang, J.J. Multitasking multi-swarm optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Wellington, New Zealand, 10–13 June 2019; pp. 1937–1944.
91. Xiao, H.; Yokoya, G.; Hatanaka, T. Multifactorial PSO-FA hybrid algorithm for multiple car design benchmark. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Bari, Italy, 6–9 October 2019; pp. 1926–1931.

92. Cheng, M.Y.; Qian, Q.; Ni, Z.W.; Zhu, X.H. Co-evolutionary particle swarm optimization for multitasking. *Pattern Recognit. Artif. Intell.* **2018**, *31*, 322–334. (In Chinese)
93. Cheng, M.Y.; Qian, Q.; Ni, Z.W.; Zhu, X.H. Information exchange particle swarm optimization for multitasking. *Pattern Recognit. Artif. Intell.* **2019**, *32*, 385–397. (In Chinese)
94. Tang, Z.D.; Gong, M.G. Adaptive multifactorial particle swarm optimisation. *Caai Trans. Intell. Technol.* **2019**, *4*, 37–46. [CrossRef]
95. Yokoya, G.; Xiao, H.; Hatanaka, T. Multifactorial optimization using artificial bee colony and its application to car structure design optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Wellington, New Zealand, 10–13 June 2019; pp. 3404–3409.
96. Xu, Z.W.; Zhang, K.; Xu, X.; He, J.J. A fireworks algorithm based on transfer spark for evolutionary multitasking. *Front. Neurorobotics* **2020**, *13*, 109. [CrossRef]
97. Cheng, M.Y.; Qian, Q.; Ni, Z.W.; Zhu, X.H. Self-organized migrating algorithm for multi-task optimization with information filtering. *J. Comput. Appl.* **2020**, in press. Available online: http://www.joca.cn/CN/10.11772/j.issn.1001-9081.2020091390 (accessed on 26 November 2020). (In Chinese).
98. Zheng, X.L.; Lei, Y.; Gong, M.G.; Tang, Z.D. Multifactorial brain storm optimization algorithm. In Proceedings of the International Conference on Bio-inspired Computing: Theories and Applications, Xi'an, China, 28–30 October 2016; pp. 47–53.
99. Lyu, C.; Shi, Y.H.; Sun, L.J. A novel multi-task optimization algorithm based on the brainstorming process. *IEEE Access* **2020**, *8*, 217134–217149. [CrossRef]
100. Osaba, E.; Ser, J.D.; Yang, X.S.; Iglesias, A.; Galvez, A. COEBA: A coevolutionary bat algorithm for discrete evolutionary multitasking. In Proceedings of the International Conference on Computational Science, Amsterdam, The Netherlands, 3–5 June 2020; pp. 244–256.
101. Chen, Q.J.; Ma, X.L.; Zhu, Z.X.; Sun, Y.W. Evolutionary multi-tasking single-objective optimization based on cooperative co-evolutionary memetic algorithm. In Proceedings of the International Conference on Computational Intelligence and Security, Hong Kong, China, 15–18 December 2017; pp. 197–201.
102. Liang, J.; Qiao, K.J.; Yuan, M.H.; Yu, K.J.; Qu, B.Y.; Ge, S.L.; Li, Y.X.; Chen, G.L. Evolutionary multi-task optimization for parameters extraction of photovoltaic models. *Energy Convers. Manag.* **2020**, *207*, 112509. [CrossRef]
103. Hashimoto, R.; Ishibuchi, H.; Masuyama, N.; Nojima, Y. Analysis of evolutionary multi-tasking as an island model. In Proceedings of the Genetic and Evolutionary Computation Conference, Kyoto, Japan, 15–19 July 2018; pp. 1894–1897.
104. Wen, Y.-W.; Ting, C.-K. Parting ways and reallocating resources in evolutionary multitasking. In Proceedings of the IEEE Congress on Evolutionary Computation, San Sebastian, Spain, 5–8 June 2017; pp. 2404–2411.
105. Zheng, X.L.; Lei, Y.; Qin, A.K.; Zhou, D.Y.; Shi, J.; Gong, M.G. Differential evolutionary multi-task optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Wellington, New Zealand, 10–13 June 2019; pp. 1914–1921.
106. Lin, J.B.; Liu, H.L.; Xue, B.; Zhang, M.J.; Gu, F.Q. Multi-objective multi-tasking optimization based on incremental learning. *IEEE Trans. Evol. Comput.* **2020**, *24*, 824–838. [CrossRef]
107. Zhou, Y.J.; Wang, T.H.; Peng, X.G. MFEA-IG: A multi-task algorithm for mobile agents path planning. In Proceedings of the IEEE Congress on Evolutionary Computation, Glasgow, UK, 19–24 July 2020; pp. 1–7.
108. Hu, H.; Zhou, Y.J.; Wang, T.H.; Peng, X.G. A multi-task algorithm for autonomous underwater vehicles 3D path planning. In Proceedings of the International Conference on Unmanned Systems, Harbin, China, 27–28 November 2020; pp. 972–977.
109. Min, A.T.W.; Ong, Y.-S.; Gupta, A.; Goh, C.K. Multiproblem surrogates: Transfer evolutionary multiobjective optimization of computationally expensive problems. *IEEE Trans. Evol. Comput.* **2019**, *23*, 15–28. [CrossRef]
110. Yin, J.; Zhu, A.M.; Zhu, Z.X.; Yu, Y.N.; Ma, X.L. Multifactorial evolutionary algorithm enhanced with cross-task search direction. In Proceedings of the IEEE Congress on Evolutionary Computation, Wellington, New Zealand, 10–13 June 2019; pp. 2244–2251.
111. Feng, Y.L.; Feng, L.; Hou, Y.Q.; Tan, K.C. Large-scale optimization via evolutionary multitasking assisted random embedding. In Proceedings of the IEEE Congress on Evolutionary Computation, Glasgow, UK, 19–24 July 2020; pp. 1–8.
112. Feng, L.; Huang, Y.X.; Zhou, L.; Zhong, J.H.; Gupta, A.; Tang, K.; Tan, K.C. Explicit evolutionary multitasking for combinatorial optimization: A case study on capacitated vehicle routing problem. *IEEE Trans. Cybern.* **2020**, in press. Available online: https://ieeexplore.ieee.org/document/9023952/ (accessed on 4 March 2020). [CrossRef] [PubMed]
113. Bali, K.K.; Gupta, A.; Feng, L.; Ong, Y.-S.; Tan, P.S. Linearized domain adaptation in evolutionary multitasking. In Proceedings of the IEEE Congress on Evolutionary Computation, San Sebastian, Spain, 5–8 June 2017; pp. 1295–1302.
114. Shang, Q.X.; Zhou, L.; Feng, L. Multi-task optimization algorithm based on denoising auto-encoder. *J. Dalian Univ. Technol.* **2019**, *59*, 417–426. (In Chinese)
115. Liang, Z.P.; Dong, H.; Liu, C.; Liang, W.Q.; Zhu, Z.X. Evolutionary multitasking for multiobjective optimization with subspace alignment and adaptive differential evolution. *IEEE Trans. Cybern.* **2020**, in press. Available online: https://ieeexplore.ieee.org/document/9123962/ (accessed on 24 June 2020). [CrossRef]
116. Xue, X.M.; Zhang, K.; Tan, K.C.; Feng, L.; Wang, J.; Chen, G.D.; Zhao, X.G.; Zhang, L.M.; Yao, J. Affine transformation-enhanced multifactorial optimization for heterogeneous problems. *IEEE Trans. Cybern.* **2020**, in press. Available online: https://ieeexplore.ieee.org/document/9295394/ (accessed on 15 December 2020). [CrossRef] [PubMed]
117. Chen, Z.F.; Zhou, Y.R.; He, X.Y.; Zhang, J. Learning task relationships in evolutionary multitasking for multiobjective continuous optimization. *IEEE Trans. Cybern.* **2020**, in press. Available online: https://ieeexplore.ieee.org/document/9262898/ (accessed on 18 November 2020). [CrossRef]

118. Xu, Q.Z.; Zhang, J.H.; Fei, R.; Li, W. Parameter analysis on multifactorial evolutionary algorithm. *J. Eng.* **2020**, *2020*, 620–625. [CrossRef]
119. Yang, C.E.; Ding, J.L.; Jin, Y.C.; Wang, C.Z.; Chai, T.Y. Multitasking multiobjective evolutionary operational indices optimization of beneficiation processes. *IEEE Trans. Autom. Sci. Eng.* **2019**, *16*, 1046–1057. [CrossRef]
120. Yang, C.E.; Ding, J.L.; Tan, K.C.; Jin, Y.C. Two-stage assortative mating for multi-objective multifactorial evolutionary optimization. In Proceedings of the IEEE 56th Annual Conference on Decision and Control, Melbourne, Australia, 12–15 December 2017; pp. 76–81.
121. Wu, T.; Bu, S.Q.; Wei, X.; Wang, G.B.; Zhou, B. Multitasking multi-objective operation optimization of integrated energy system considering biogas-solar-wind renewables. *Energy Convers. Manag.* **2021**, *229*, 113736. [CrossRef]
122. Liaw, R.-T.; Ting, C.-K. Evolutionary many-tasking based on biocoenosis through symbiosis: A framework and benchmark problems. In Proceedings of the IEEE Congress on Evolutionary Computation, San Sebastian, Spain, 5–8 June 2017; pp. 2266–2273.
123. Liaw, R.-T.; Ting, C.-K. Evolution of biocoenosis through symbiosis with fitness approximation formany-tasking optimization. *Memetic Comput.* **2020**, *12*, 399–417. [CrossRef]
124. Binh, H.T.T.; Tuan, N.Q.; Long, D.C.T. A multi-objective multi-factorial evolutionary algorithm with reference-point-based approach. In Proceedings of the IEEE Congress on Evolutionary Computation, Wellington, New Zealand, 10–13 June 2019; pp. 2824–2831.
125. Zheng, X.L.; Qin, A.K.; Gong, M.G.; Zhou, D.Y. Self-regulated evolutionary multi-task optimization. *IEEE Trans. Evol. Comput.* **2020**, *24*, 16–28. [CrossRef]
126. Osaba, E.; Martinez, A.D.; Galvez, A.; Iglesias, A.; Ser, J.D. dMFEA-II: An adaptive multifactorial evolutionary algorithm for permutation-based discrete optimization problems. In Proceedings of the Genetic and Evolutionary Computation Conference, Cancún, Mexico, 8–12 July 2020; pp. 1690–1696.
127. Gong, M.G.; Tang, Z.D.; Li, H.; Zhang, J. Evolutionary multitasking with dynamic resource allocating strategy. *IEEE Trans. Evol. Comput.* **2019**, *23*, 858–869. [CrossRef]
128. Yao, S.S.; Dong, Z.M.; Wang, X.P.; Ren, L. A Multiobjective multifactorial optimization algorithm based on decomposition and dynamic resource allocation strategy. *Inf. Sci.* **2020**, *511*, 18–35. [CrossRef]
129. Chen, Q.J.; Ma, X.L.; Sun, Y.W.; Zhu, Z.X. Adaptive memetic algorithm based evolutionary multi-tasking single-objective optimization. In Proceedings of the Asia-Pacific Conference on Simulated Evolution and Learning, Shenzhen, China, 10–13 November 2017; pp. 462–472.
130. Tang, J.; Chen, Y.K.; Deng, Z.X.; Xiang, Y.P.; Joy, C.P. A group-based approach to improve multifactorial evolutionary algorithm. In Proceedings of the International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 13–19 July 2018; pp. 3870–3876.
131. Tang, Z.D.; Gong, M.G.; Jiang, F.L.; Li, H.; Wu, Y. Multipopulation optimization for multitask optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Wellington, New Zealand, 10–13 June 2019; pp. 1906–1913.
132. Jin, C.; Tsai, P.-W.; Qin, A.K. A study on knowledge reuse strategies in multitasking differential evolution. In Proceedings of the IEEE Congress on Evolutionary Computation, Wellington, New Zealand, 10–13 June 2019; pp. 1564–1571.
133. Ma, X.L.; Chen, Q.J.; Yu, Y.N.; Sun, Y.W.; Ma, L.J.; Zhu, Z.X. A two-level transfer learning algorithm for evolutionary multitasking. *Front. Neurosci.* **2020**, *13*, 1408. [CrossRef]
134. Xie, T.; Gong, M.G.; Tang, Z.D.; Lei, Y.; Liu, J.; Wang, Z. Enhancing evolutionary multifactorial optimization based on particle swarm optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, 24–29 July 2016; pp. 1658–1665.
135. Da, B.S.; Ong, Y.-S.; Feng, L.; Qin, A.K.; Gupta, A.; Zhu, Z.X.; Ting, C.-K.; Tang, K.; Yao, X. *Evolutionary Multitasking for Single-Objective Continuous Optimization: Benchmark Problems, Performance Metric and Baseline Results*; Technical Report; Nanyang Technological University: Singapore, 2016.
136. Zhou, L.; Feng, L.; Zhong, J.H.; Zhu, Z.X.; Da, B.S.; Wu, Z. A study of similarity measure between tasks for multifactorial evolutionary algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference, Kyoto, Japan, 15–19 July 2018; pp. 229–230.
137. Gupta, A.; Ong, Y.-S.; Da, B.S.; Feng, L.; Handoko, S.D. Landscape synergy in evolutionary multitasking. In Proceedings of the IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, 24–29 July 2016; pp. 3076–3083.
138. Nguyen, T.B.; Browne, W.N.; Zhang, M.J. Relatedness measures to aid the transfer of building blocks among multiple tasks. In Proceedings of the Genetic and Evolutionary Computation Conference, Cancún, Mexico, 8–12 July 2020; pp. 377–385.
139. Sagarna, R.; Ong, Y.-S. Concurrently searching branches in software tests generation through multitask evolution. In Proceedings of the IEEE Symposium Series on Computational Intelligence, Athens, Greece, 6–9 December 2016; pp. 1–8.
140. Scott, E.O.; De Jong, K.A. Automating knowledge transfer with multi-task optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Wellington, New Zealand, 10–13 June 2019; pp. 2252–2259.
141. Chen, Y.L.; Zhong, J.H.; Feng, L.; Zhang, J. An adaptive archive-based evolutionary framework for many-task optimization. *IEEE Trans. Emerg. Top. Comput. Intell.* **2020**, *4*, 369–384. [CrossRef]
142. Shang, Q.; Zhang, L.; Feng, L.; Hou, Y.; Zhong, J.; Gupta, A.; Tan, K.C.; Liu, H.L. A preliminary study of adaptive task selection in explicit evolutionary many-tasking. In Proceedings of the IEEE Congress on Evolutionary Computation, Wellington, New Zealand, 10–13 June 2019; pp. 2153–2159.

143. Xu, Q.Z.; Tian, B.L.; Wang, L.; Sun, Q.; Zou, F. An effective variable transfer strategy in multitasking optimization. In Proceedings of the Genetic and Evolutionary Computation Conference, Cancún, Mexico, 8–12 July 2020; pp. 59–60.
144. Xu, Q.Z.; Wang, L.; Yang, J.G.; Wang, N.; Fei, R.; Sun, Q. An effective variable transformation strategy in multitasking evolutionary algorithms. *Complexity* **2020**, *2020*, 8815117. [CrossRef]
145. Zhang, D.Q.; Jiang, M.Y. Hetero-dimensional multitask neuroevolution for chaotic time series prediction. *IEEE Access* **2020**, *8*, 123135–123150. [CrossRef]
146. Wang, L.; Sun, Q.; Xu, Q.Z.; Tian, B.L.; Li, W. On the order of variables for multitasking optimization. In Proceedings of the Genetic and Evolutionary Computation Conference, Cancún, Mexico, 8–12 July 2020; pp. 57–58.
147. Wang, L.; Sun, Q.; Xu, Q.Z.; Li, W.; Jiang, Q.Y. Analysis of multitasking evolutionary algorithms under the order of solution variables. *Complexity* **2020**, *2020*, 4609489. [CrossRef]
148. Zhou, L.; Feng, L.; Tan, K.C.; Zhong, J.H.; Zhu, Z.X.; Liu, K.; Chen, C. Toward adaptive knowledge transfer in multifactorial evolutionary computation. *IEEE Trans. Cybern.* **2020**, in press. Available online: https://ieeexplore.ieee.org/document/9027113/ (accessed on 6 March 2020). in press. [CrossRef]
149. Zhou, L.; Feng, L.; Liu, K.; Chen, C.; Deng, S.J.; Xiang, T.; Jiang, S.W. Towards effective mutation for knowledge transfer in multifactorial differential evolution. In Proceedings of the IEEE Congress on Evolutionary Computation, Wellington, New Zealand, 10–13 June 2019; pp. 1541–1547.
150. Gong, D.W.; Xu, B.; Zhang, Y.; Guo, Y.N.; Yang, S.X. A similarity-based cooperative co-evolutionary algorithm for dynamic interval multiobjective optimization problems. *IEEE Trans. Evol. Comput.* **2020**, *24*, 142–156. [CrossRef]
151. Gong, D.W.; Sun, J.; Miao, Z. A set-based genetic algorithm for interval many-objective optimization problems. *IEEE Trans. Evol. Comput.* **2018**, *22*, 47–60. [CrossRef]
152. Więckowski, J.; Kizielewicz, B.; Kołodziejczyk, J. The search of the optimal preference values of the characteristic objects by using particle swarm optimization in the uncertain environment. In Proceedings of the 12th KES International Conference on Intelligent Decision Technologies, Split, Croatia, 17–19 June 2020; pp. 353–363.
153. Burke, E.; Kendall, G.; Newall, J.; Hart, E.; Ross, P.; Schulenburg, S. Hyper-heuristics: An emerging direction in modern search technology. In *Handbook of Metaheuristics*; Glover, F.W., Kochenberger, G.A., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; pp. 457–474.
154. Pillay, N.; Qu, R. *Hyper-Heuristics: Theory and Applications*; Springer: Berlin/Heidelberg, Germany, 2018.
155. Więckowski, J.; Kizielewicz, B.; Kołodziejczyk, J. Application of hill climbing algorithm in determining the characteristic objects preferences based on the reference set of alternatives. In Proceedings of the 12th KES International Conference on Intelligent Decision Technologies, Split, Croatia, 17–19 June 2020; pp. 341–351.
156. Więckowski, J.; Kizielewicz, B.; Kołodziejczyk, J. Finding an approximate global optimum of characteristic objects preferences by using simulated annealing. In Proceedings of the 12th KES International Conference on Intelligent Decision Technologies, Split, Croatia, 17–19 June 2020; Springer: Singapore, 2020; pp. 365–375.
157. Zhou, Z.F.; Ma, X.L.; Liang, Z.P.; Zhu, Z.X. Multi-objective multi-factorial memetic algorithm based on bone route and large neighborhood local search for VRPTW. In Proceedings of the IEEE Congress on Evolutionary Computation, Glasgow, UK, 19–24 July 2020; pp. 1–8.
158. Huang, L.Y.; Feng, L.; Wang, H.D.; Hou, Y.Q.; Liu, K.; Chen, C. A preliminary study of improving evolutionary multi-objective optimization via knowledge transfer from single-objective problems. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Toronto, ON, Canada, 11–14 October 2020; pp. 1552–1559.
159. Zheng, Y.J.; Zhu, Z.X.; Qi, Y.T.; Wang, L.; Ma, X.L. Multi-objective multifactorial evolutionary algorithm enhanced with the weighting helper-task. In Proceedings of the International Conference on Industrial Artificial Intelligence, Shenyang, China, 23–25 October 2020; pp. 1–6.
160. Yu, Y.N.; Zhu, A.M.; Zhu, Z.X.; Lin, Q.Z.; Yin, J.; Ma, X.L. Multifactorial differential evolution with opposition-based learning for multi-tasking optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Wellington, New Zealand, 10–13 June 2019; pp. 1898–1905.
161. Yao, S.S.; Dong, Z.M.; Wang, X.P. A multiobjective multifactorial evolutionary algorithm based on decomposition. *Control Decis.* **2021**, *36*, 637–644. (In Chinese)
162. Mo, J.J.; Fan, Z.; Li, W.J.; Fang, Y.; You, Y.G.; Cai, X.Y. (2017) Multi-factorial evolutionary algorithm based on M2M decomposition. In Proceedings of the Asia-Pacific Conference on Simulated Evolution and Learning, Shenzhen, China, 10–13 November 2017; pp. 134–144.
163. Liao, P.; Sun, C.L.; Zhang, G.C.; Jin, Y.C. Multi-surrogate multi-tasking optimization of expensive problems. *Knowl.-Based Syst.* **2020**, *205*, 106262. [CrossRef]
164. Binh, H.T.T.; Thanh, P.D.; Trung, T.B.; Thanh, L.C.; Phong, L.M.H.; Swami, A.; Lam, B.T. A multifactorial optimization paradigm for linkage tree genetic algorithm. *Inf. Sci.* **2020**, *540*, 325–344.
165. Park, J.; Mei, Y.; Nguyen, S.; Chen, G.; Zhang, M.J. Evolutionary multitask optimisation for dynamic job shop scheduling using niched genetic programming. In Proceedings of the Australasian Joint Conference on Artificial Intelligence, Wellington, New Zealand, 11–14 December 2018; pp. 739–751.
166. Rauniyar, A.; Nath, R.; Muhuri, P.K. Multi-factorial evolutionary algorithm based novel solution approach for multi-objective pollution routing problem. *Comput. Ind. Eng.* **2019**, *130*, 757–771. [CrossRef]

167. Karunakaran, D.; Mei, Y.; Zhang, M.J. Multitasking genetic programming for stochastic team orienteering problem with time windows. In Proceedings of the IEEE Symposium Series on Computational Intelligence, Xiamen, China, 6–9 December 2019; pp. 1598–1605.
168. Zhuang, Z.Y.; Wei, C.; Li, B.; Xu, P.; Guo, Y.F.; Ren, J.C. Performance prediction model based on multi-task learning and co-evolutionary strategy for ground source heat pump system. *IEEE Access* **2019**, *7*, 117925–117933. [CrossRef]
169. Shen, F.; Liu, J.; Wu, K. Evolutionary multitasking fuzzy cognitive map learning. *Knowl. -Based Syst.* **2019**, *192*, 105294. [CrossRef]
170. Zhang, B.Y.; Qin, A.K.; Pan, H.; Sellis, T. A novel DNN training framework via data sampling and multi-task optimization. In Proceedings of the International Joint Conference on Neural Networks, Glasgow, UK, 19–24 July 2020; pp. 1–8.
171. Martinez, A.D.; Osaba, E.; Ser, J.D.; Herrera, F. Simultaneously evolving deep reinforcement learning models using multifactorial optimization. In Proceedings of the IEEE Conference on Evolutionary Computation, Glasgow, UK, 19–24 July 2020; pp. 1–8.
172. Wei, T.Y.; Zhong, J.H. A preliminary study of knowledge transfer in multi-classification using gene expression programming. *Front. Neurosci.* **2020**, *13*, 1396. [CrossRef]
173. Chen, K.; Xue, B.; Zhang, M.J.; Zhou, F.Y. An evolutionary multitasking-based feature selection method for high-dimensional classification. *IEEE Trans. Cybern.* **2020**. Available online: https://ieeexplore.ieee.org/document/9311803/ (accessed on 31 December 2020).
174. Tang, Z.D.; Gong, M.G.; Zhang, M.Y. Evolutionary multi-task learning for modular extremal learning machine. In Proceedings of the IEEE Congress on Evolutionary Computation, San Sebastian, Spain, 5–8 June 2017; pp. 474–479.
175. Li, H.; Ong, Y.-S.; Gong, M.G.; Wang, Z.K. Evolutionary multitasking sparse reconstruction: Framework and case study. *IEEE Trans. Evol. Comput.* **2019**, *23*, 733–747. [CrossRef]
176. Zhao, Y.Z.; Li, H.; Wu, Y.; Wang, S.F.; Gong, M.G. Endmember selection of hyperspectral images based on evolutionary multitask. In Proceedings of the IEEE Congress on Evolutionary Computation, Glasgow, UK, 19–24 July 2020; pp. 1–7.
177. Sampath, L.P.M.I.; Gupta, A.; Ong, Y.-S.; Gooi, H.B. Evolutionary multitasking to support optimal power flow under rapid load variations. *South. Power Syst. Technol.* **2017**, *11*, 103–114.
178. Liu, J.W.; Li, P.L.; Wang, G.B.; Zha, Y.X.; Peng, J.C.; Xu, G. A multitasking electric power dispatch approach with multi-objective multifactorial optimization algorithm. *IEEE Access* **2020**, *8*, 155902–155910. [CrossRef]
179. Bao, L.; Qi, Y.T.; Shen, M.Q.; Bu, X.X.; Yu, J.S.; Li, Q.; Chen, P. An evolutionary multitasking algorithm for cloud computing service composition. In Proceedings of the World Congress on Services, Seattle, WA, USA, 25–30 June 2018; pp. 130–144.
180. Singh, D.; Sisodia, D.S.; Singh, P. Compositional framework for multitask learning in the identification of cleavage sites of HIV-1 protease. *J. Biomed. Inform.* **2020**, *102*, 103376. [CrossRef]
181. Sinha, A.; Malo, P.; Deb, K. Unconstrained scalable test problems for single-objective bilevel optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Brisbane, Australia, 10–15 June 2012; pp. 1–8.
182. Ruan, G.; Minku, L.L.; Menzel, S.; Sendhoff, B.; Yao, X. When and how to transfer knowledge in dynamic multi-objective optimization. In Proceedings of the IEEE Symposium Series on Computational Intelligence, Xiamen, China, 6–9 December 2019; pp. 2034–2041.
183. Kohira, T.; Akira, O.; Kemmotsu, H.; Tatsukawa, T. Proposal of benchmark problem based on real-world car structure design optimization. In Proceedings of the Genetic and Evolutionary Computation Conference, Kyoto, Japan, 15–19 July 2018; pp. 183–184.
184. Xu, Q.Z.; Yang, H.; Wang, N.; Wu, G.H.; Jiang, Q.Y. Recent advances in multifactorial evolutionary algorithm. *Comput. Eng. Appl.* **2018**, *54*, 15–20. (In Chinese)
185. Hao, G.-S.; Wang, G.-G.; Zhang, Z.-J.; Zou, D.-X. Optimization of the high order problems in evolutionary algorithms: An application of transfer learning. *Int. J. Wirel. Mob. Comput.* **2018**, *14*, 56–63. [CrossRef]
186. Wang, G.-G.; Tan, Y. Improving metaheuristic algorithms with information feedback models. *IEEE Trans. Cybern.* **2019**, *49*, 542–555. [CrossRef]
187. Yuan, Y.; Ong, Y.-S.; Feng, L.; Qin, A.K.; Gupta, A.; Da, B.S.; Zhang, Q.F.; Tan, K.C.; Jin, Y.C.; Ishibuchi, H. *Evolutionary Multitasking for Multiobjective Continuous Optimization: Benchmark Problems, Performance Metrics and Baseline Results*; Technical Report; Nanyang Technological University: Singapore, 2016.
188. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
189. Sands, T. Comparison and interpretation methods for predictive control of mechanics. *Algorithms* **2019**, *12*, 232. [CrossRef]
190. Sałabun, W.; Wątróbski, J.; Shekhovtsov, A. Are MCDA methods benchmarkable? A comparative study of TOPSIS, VIKOR, COPRAS, and PROMETHEE II methods. *Symmetry* **2020**, *12*, 1549. [CrossRef]
191. Jiang, S.W.; Xu, C.; Gupta, A.; Feng, L.; Ong, Y.-S.; Zhang, A.N.; Tan, P.S. Complex and intelligent systems in manufacturing. *IEEE Potentials* **2016**, *35*, 23–28. [CrossRef]
192. Sands, T. Development of deterministic artificial intelligence for unmanned underwater vehicles (UUV). *J. Mar. Sci. Eng.* **2020**, *8*, 578. [CrossRef]

mathematics

MDPI

Article

Using Cuckoo Search Algorithm with *Q*-Learning and Genetic Operation to Solve the Problem of Logistics Distribution Center Location

Juan Li [1,2], Dan-dan Xiao [1], Hong Lei [2], Ting Zhang [1] and Tian Tian [3,*]

[1] School of Information Engineering, Wuhan Technology and Business University, Wuhan 430065, China; looj@whu.edu.cn (J.L.); xiaodandan@wtbu.edu.cn (D.-d.X.); 20040808005@wtbu.edu.cn (T.Z.)
[2] School of Artificial Intelligence, Wuchang University of Technology, Wuhan 430223, China; 120150508@wut.edu.cn
[3] School of Computer Science and Technology, Shandong Jianzhu University, Jinan 250101, China
* Correspondence: tiantian@sdjzu.edu.cn

Received: 14 December 2019; Accepted: 17 January 2020; Published: 21 January 2020

Abstract: Cuckoo search (CS) algorithm is a novel swarm intelligence optimization algorithm, which is successfully applied to solve some optimization problems. However, it has some disadvantages, as it is easily trapped in local optimal solutions. Therefore, in this work, a new CS extension with *Q*-Learning step size and genetic operator, namely dynamic step size cuckoo search algorithm (DMQL-CS), is proposed. Step size control strategy is considered as action in DMQL-CS algorithm, which is used to examine the individual multi-step evolution effect and learn the individual optimal step size by calculating the *Q* function value. Furthermore, genetic operators are added to DMQL-CS algorithm. Crossover and mutation operations expand search area of the population and improve the diversity of the population. Comparing with various CS algorithms and variants of differential evolution (DE), the results demonstrate that the DMQL-CS algorithm is a competitive swarm algorithm. In addition, the DMQL-CS algorithm was applied to solve the problem of logistics distribution center location. The effectiveness of the proposed method was verified by comparing with cuckoo search (CS), improved cuckoo search algorithm (ICS), modified chaos-enhanced cuckoo search algorithm (CCS), and immune genetic algorithm (IGA) for both 6 and 10 distribution centers.

Keywords: global optimization; cuckoo search algorithm; *Q*-learning; mutation; self-adaptive step size

1. Introduction

Optimization problems have been one of the most important research topics in recent years. They exist in many domains, such as scheduling [1,2], image processing [3–6], feature selection [7–9] and detection [10], path planning [11,12], feature selection [13], cyber-physical social system [14,15], texture discrimination [16], saliency detection [17], classification [18,19], object extraction [20], shape design [21], big data and large-scale optimization [22,23], multi-objective optimization [24], knapsack problem [25–27], fault diagnosis [28–30], and test-sheet composition [31]. Metaheuristic algorithms [32], a theoretical tool, are based on nature-inspired ideas, which have been extensively used to solve highly non-linear complex multi-objective optimization problems [33–35]. Several popular metaheuristics with a stochastic nature are compared in some studies [36–38] with deterministic Lipschitz methods by using operational zones. Most of these metaheuristics methods are inspired by natural or physical processes, such as bat algorithm (BA) [39], biogeography-based optimization (BBO) [40], ant colony optimization (ACO) [41], earthworm optimization algorithm (EWA) [42], elephant herding optimization (EHO) [43,44], moth search (MS) algorithm [45], firefly algorithm (FA) [46], artificial bee

colony (ABC) [47–49], harmony search (HS) [50,51], monarch butterfly optimization (MBO) [52,53], particle swarm optimization (PSO) [54,55], genetic programming [56], krill herd (KH) [57–63], immune genetic algorithm (IGA) [64], and cuckoo search (CS) [65–69].

Yang and Deb [69] proposed a metaheuristic optimization method named CS algorithm, which is inspired by smart incubation behavior of a type of birds called cuckoos in nature.

CS performs local search well in most cases, but sometimes it cannot escape from local optima, which restricts its ability to carry out full search globally. To enhance the ability of CS, Mlakar et al. [70] proposed a novel hybrid self-adaptively CS algorithm adding three features: a self-adaptively of cuckoo search control parameters, a linear population reduction, and a balancing of the exploration search strategies. Li et al. [71] enhanced the exploitation ability of the cuckoo search algorithm by using an orthogonal learning strategy. An improved discrete version of CS was presented by Ouaarab et al. [72].

On the other hand, most researchers agree that the performance of algorithms can be improved by using learning techniques. For example, Wang et al. [73] presented a new method to enhance learning speed and improved final performance, which directly tuned the Q-values to affect the action selection policy. Alex et al. [74] presented a new evolutionary cooperative learning scheme that is able to solve function approximation and classification problems, improving accuracy and generalization capabilities. A new CS algorithm named snap-drift cuckoo search (SDCS) was presented by Hojjat et al. [75]. In SDCS, a snap-drift learning strategy is employed to improve search operators. The snap-drift learning strategy provides an online trade-off between local and global search via two snap and drift modes.

Although much effort has been made to enhance the performance of CS, many of the variants fail to improve the performance of CS algorithm on certain complicated problems. Furthermore, there are few studies on optimizing the parameters of CS algorithm by using learning strategy. In this paper, we present an improved CS algorithm called dynamic step size cuckoo search algorithm (DMQL-CS) that adopts strategies with Q-Learning and genetic operator. Step size strategy of the traditional CS focused only on examining the individual fitness value based on the one-step evolution effect of individual, but ignored the evaluation of step size from the multi-step evolution effect, which is not conducive to the evolution of the algorithm. We use Q-Learning method to optimize the step size, in which the most appropriate step size control strategies are retained for the next generation. At the same time, their weights are adaptively adjusted by using learning rate, which is used to guide individuals to search for a better solution at the next evolution. In addition, crossover operation and mutation operation are added into the DMQL-CS algorithm to accelerate the convergence speed of the algorithm and expand the diversity of the population.

The present manuscript differs from other similar work insofar as the advantage of learning based on Q-Learning and genetic operators. Q-Learning considers the multi-step evolution effect of individual such that the most appropriate step size control strategies are retained for the next generation. For the proposed DMQL-CS approach, the outstanding work of the paper is mainly listed in the following two aspects:

(1) In the DMQL-CS algorithm, the step size strategy is considered as an action which applies multiple step control strategies (linear decreasing strategy, non-linear decreasing strategy, and adaptively step-size strategy). In the DMQL-CS algorithm, according to multi-step effect of individual for a few steps forward, the optimal step size control strategy is learned. During each learning evolution step size, finally, the optimal individual and corresponding optimal step size strategy are derived by calculating the Q function value. The current individual continues to evolve through the step size obtained, which increases the adaptability of individual evolution.

(2) The research introduces two genetic operators, crossover and mutation, into the DMQL-CS algorithm, intended for accelerating convergence. During crossover and mutation process, chromosomes are divided into pairs according to certain probability. We introduce the specifically designed crossover operation into problem of logistics distribution center location in this paper, which determines the performance of the algorithm to some extent. To improve the search ability

of the CS algorithm, numerous strategies have been designed to adjust the crossover rate. In this work, a self-adaptive scheme is used to adjust the crossover rate. Genetic operators expand the search area of the population to improve the exploration and maintain the diversity of the population, which also helps to improve the exploration of the population of learners.

Finally, the DMQL-CS method was tested on 15 benchmark functions, CEC 2013 test suite, and the problem of logistics distribution center location. The experimental results compared with those of other approaches demonstrated the superiority of the proposed strategy. A series of simulation experiments showed that DMQL-CS performs more accurately and efficiently than other evolutionary methods in terms of the quality of the solution and convergence rate.

The remainder of this paper is organized as follows. In Section 2, the related work on cuckoo search is presented. Section 3 presents cuckoo search. The proposed DMQL-CS algorithm, including *Q*-Learning model, step size control model with *Q*-Learning, and genetic operator, is described in Section 4. The comparison with other methods, through 15 functions, CEC 2013 test suite, and the problem of logistics distribution center location, is given in Section 5. Finally, Section 6 concludes this paper and points out some future research directions.

2. Related Work

CS algorithm is capable of finding the best solutions by continuously using new and potentially better solution to replace a not-so-good cuckoo in the population, and it has been applied successfully to diverse fields. Recently, many CS variants have been developed to improve the performance of the CS algorithm. These variants can be generally divided into four categories: (1) parameter control [70]; (2) novel learning schemes [76]; (3) hybrid methods with other algorithm [74]; and (4) local search operator [77].

Due to the important influence of control parameters for the performance, much meaningful work has been done on the control parameter settings of CS algorithm. Initially, step size parameter control was investigated to improve the performance of CS algorithms. For instance, aiming at the faults that Cuckoo Search algorithm cannot acquire exact solutions and converges slowly in the later period, Ma et al. [78] proposed a self-adaptively step size adjustment cuckoo search algorithm (ASCS), which is an adaptively adjusted step size by using the distance between cuckoo nest location and the optimal nest location, which speeds up CS algorithm speed and improves the calculation accuracy. To balance the exploration and exploitation, Li and Yin [79] introduced two mutation rules and combined these two rules using a linear decreasing probability. Then, an adaptive parameter adjustment strategy was developed according to the relative success number of two newly added parameters in the previous iteration. Comparison results of the proposed algorithm show that this scheme is better than other algorithms. Two important factors, speed factor and aggregation factor, were defined by Yang et al. [80]. Then, according to these two factors, the step size and discovery probability were regulated. Experimental results show that the CS with improved step size and discovery probability has strong competitiveness in tackling numerical optimization problems. Li et al. [79] proposed the self-adaptive parameter CS algorithm, which uses two new mutation rules based on the rand and best individuals among the entire population. The self-adaptive parameter is set as a uniform random value based on the relative success number of the two new proposed parameters in the previous period, which enhance diversity of the population. Experimental results show that the proposed method performs better than twelve algorithms from the literature.

Li et al. [65] proposed an enhanced CS algorithm called dynamic CS with Taguchi opposition-based search and dynamic evaluation. The Taguchi search strategy provided random generalized learning based on opposing relationships to enhance the exploration ability of the algorithm. The dynamic evaluation strategy reduced the number of function evaluations, and accelerated the convergence property. Statistical comparisons of experimental results showed that the proposed algorithm makes an appropriate trade-off between exploration and exploitation. Li et al. [81] proposed a new cuckoo search algorithm extension based on self-adaptive knowledge learning, in which a learning model

with individual history knowledge and population knowledge is introduced into the CS algorithm. Individuals constantly adjust their position according to historical knowledge and communicate in the optimization process. Statistical comparisons of the experimental results showed that the proposed algorithm is a competitive new type of algorithm. Hojjat et al. [75] presented a new CS algorithm, called snap-drift cuckoo search (SDCS), which first employs a learning strategy and then considers improved search operators. The snap-drift learning strategy provides an online trade-off between local and global search via two snap and drift modes. SDCS tends to increase global search to prevent algorithm of being trapped in local minima via snap mode and reinforces the local search to enhance the convergence rate via drift mode. Statistical comparisons of experimental results showed that SDCS is superior to modified CS algorithms in terms of convergence speed and robustness.

According to the rand and best individuals among the entire population, Cheng et al. [82] proposed an ensemble CS variant in which three different cuckoo search algorithms coexist in the entire search process, which compete to produce better offspring for numerical optimization. Then, an external archive is introduced to further maintain population diversity. Statistical comparisons of experimental results showed that the improved CS variant is superior to modified CS algorithms in terms of convergence speed and robustness. Wen et al. [83] proposed a new hybrid algorithm based on grey wolf optimizer and cuckoo search (GWOCS), which was developed to extract the parameters of different PV cell models with the experimental data under different operating conditions. Zhang et al. [84] proposed an ensemble CS variant that divides the population into two subgroups and adopts CS and DE for these two subgroups independently. These two subgroups can exchange useful information by division. These two algorithms can utilize each other's advantages to complement their shortcomings, thus balancing the quality of solution and the computation consumption. Zhang et al. [85] devised a hybridization of CS and covariance matrix adaption evolution strategy (CMA_ES) to improve performance for the different optimization problems. Computational results demonstrate that the proposed algorithm outperforms other competitor algorithms. Tang et al. [86] introduced Gaussian distribution, Cauchy distribution, Levy distribution, and Uniform distribution, improving the performance of cuckoo search algorithm by the method of pair combination. Simulation results show that the hybrid distribution with Cauchy distribution and Levy distribution can make the CS algorithm perform better.

With respect to applications, CS has been extensively applied to many domains, such as neural networks [87], image processing [88], nonlinear systems [89,90], network structural optimization [91], agriculture optimization [92], engineering optimization [93], and scheduling [94]. These applications indicate that CS algorithm is an effective and efficient optimizer for solving some real-world problems.

3. Cuckoo Search

The cuckoo search algorithm [69] is a stochastic optimization algorithm that models brood parasitism of cuckoo birds. The algorithm is based on the obligate brood parasitic behavior found in some cuckoo nests by combining a model of this behavior with the principles of Lévy flights, which discard worst solutions and generate new ones after some certain iteration.

According to the mentioned characteristics, CS can be expressed as three idealized rules:

(1) Each cuckoo lays one egg at a time, and places it in a randomly chosen nest.
(2) The best nests with the highest-quality eggs (solutions) will be carried over to the next generations.
(3) The number of available host nests is fixed, and the alien egg is discovered by the host bird with the probability $p_a \in [0, 1]$. If the alien egg is discovered, the nest is abandoned and a new nest is built in a new location.

The CS algorithm is equiponderant to the integration of Lévy flights. The position of the ith nest is indicated by using D-dimensional vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{id}), 1 \le i \le n$; a Lévy flight is performed:

$$X_i^{t+1} = x_i^t + a \otimes levy(\lambda)\ (i = 1, 2, \ldots, n), \tag{1}$$

$$a = a_0 \otimes (x_j^t - x_i^t) \pm \tag{2}$$

where $\alpha > 0$ is the step size that is used to control the range of the random search, which should be related to the scales of the problem of interests, and step size information is more useful can be computed by Equation (2). The product $\otimes$ means entry-wise multiplications. x_i^t and x_j^t are two different solutions selected randomly. A new solution with the same number of cuckoos is generated after partial solutions are discarded. $levy(\lambda)$ with the random walk can be expressed in terms of a simple power-law equation.

$$levy(\beta) \sim \mu = t^{-1-\beta},\ 0 < \beta \leq 2 \tag{3}$$

where μ and t are two random numbers following the normal distribution and β often takes a fixed value of 1.5.

$$levy(\beta) \sim \frac{\phi \times \mu}{|v|^{1/\beta}} \tag{4}$$

$$\phi = \left[\frac{\Gamma(1+\beta) \times \sin(\frac{\pi \times \beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{\frac{\beta-1}{2}}} \right]^{1/\beta} \tag{5}$$

where Γ is gamma function. μ and v are random numbers drawn from a normal distribution with mean of 0 and standard deviation of 1, which have an infinite variance with an infinite mean. Here, the consecutive jumps/steps of a cuckoo essentially form a random walk process that obeys a power-law step length distribution with a heavy tail. In Lévy flights random walk component, the new solution X_i is generated through Equation (6).

$$X_{g+1,i} = X_{g,i} + \alpha_0 \frac{\phi \times \mu}{|v|^{1/\beta}} (X_{g,i-} X_{g,best}) \tag{6}$$

where $X_{g,best}$ represents the best solution obtained thus far and α_0 is a scaling factor. The Lévy distribution is a process of random walk; after a series of smaller steps, Lévy flights can suddenly obtain a relatively larger step size. Lévy distribution is implemented at the initial stage of algorithm, which helps to jump out of the local optimum.

$$X_i^{t+1} = x_i^t + r(X_m^t - X_n^t) \tag{7}$$

where X_m^t and X_n^t are random solutions at the tth generation. r generates a random number between -1 and 1. The basic steps of the CS algorithm are summarized in Algorithm 1.

Algorithm 1 CS Algorithm.

(1) randomly initialize population of n host nests
(2) calculate fitness value for each solution in each nest
(3) **while** (stopping criterion is not meet do)
(4) Generate x_i^{t+1} as new solution by using Lévy flights;
(5) Choose candidate solution x_i^t;
(6) **if** $f(x_i^t) > f(x_i^{t+1})$
(7) Replace x_i^t with new solution x_i^{t+1};
(8) **end if**
(9) Throw out a fraction (p_a) of worst nests;
(10) Generate solution k_i^{t+1} using Equation (3);
(11) **if** $f(x_i^t) > f(x_i^{t+1})$
(12) Replace x_i^t with new solution x_i^{t+1};
(13) **end if**
(14) Rank the solution and find the current best.
(15) **end while**

4. Cuckoo Search Algorithm with *Q*-Learning and Genetic Operations

4.1. Q-Learning Model

Q-Learning model, a milestone in reinforcement learning research, is an enhanced learning method that is not constrained by the problem model. The optimal policy of *Q*-Learning is generated by executing the action with the highest expected *Q*-values, which is the action of maximizing the cumulative benefits with a discount. Control strategy of the optimal step size can be transformed into the optimal action for the agent. The *Q* function is defined as discounted. In general, the environment is the current state in which the agent makes decisions. The agent includes a set of feasible actions which affect both next state and reward. In fact, the *Q*-Learning is a mapping from state–action to prediction. The output for state vector s and action a are denoted by *Q*-value $Q(s,a)$:

$$Q(s_t,a_t) \leftarrow (1-a)Q(s_t,a_t) + a\left[r_{t+1} + \gamma \underset{a_{t+1}}{max} Q(s_{t+1},a_{t+1})\right] \tag{8}$$

where $Q(s_t,a_t)$ represents the cumulative reward of action in the state of s at time t. $Q(s_{t+1},a_{t+1})$ indicates the cumulative reward of action in the state vector s at time $t+1$. r_{t+1} is the reward received for the action a at time $t+1$. When s_{t+1} is terminal, $Q(s_{t+1},a_{t+1})$ goes to zero, where a and γ represent learning factors and discount factors, respectively ($0 < a < 1$, $0 \le \gamma < 1$). γ determines the impact of lagging returns on optimal action. *Q*-Learning provides strong proof of convergence. The *Q* value will converge with probability 1 to *Q* when each state–action pair is repeatedly visited. The error of *Q* (s, a) must be reduced by γ whenever it is updated. When each state–action pair is visited infinitely, the estimates of $Q_n(s,a)$ converge to real values of $Q(s,a)$ as $n \to \infty$.

4.2. Step Size Control Model by Using Q-Learning

In CS algorithm, the most important parameter is step size scaling factor with the typical characteristics of Lévy flight, in addition to the population size, the number of iterations, and the probability of discovery. Step size scaling factor is as suitable action that is selected to control an individual search process. The accuracy of selected parameter can be improved by predicting before making an action decision. When an individual selects an action, the advantages and disadvantages of various actions can be evaluated by the multi-step effect of individual. *Q*-Learning is helpful to learn the optimal step size control strategy and transform optimal step size control strategy into optimal action selected of agent.

During the iteration of CS algorithm, the fixed step size strategy cannot meet the dynamic requirements of the algorithm. Considering the aforementioned facts, at the later stage of the CS algorithm, we add three step size control methods in the iterative process: (1) Dynamic linear decreasing strategy (L1) is defined by Equation (9). (2) Dynamic non-linear decreasing strategy (L2) is defined by Equation (10). (3) Adaptive step-size strategy (L3) is defined by Equation (11). Each individual obtains the optimal step size control strategy via learning multiple steps forward, thus becomes close to the optimal solution. Therefore, we try to evaluate the step size control strategy by using multi-step evolution method, which increases the adaptability of individual evolution and improves the performance of the algorithm. The current best step size control strategy is selected to execute the next iteration by using *Q*-Learning method.

$$a = (a_1 - a_0) \times (t_{\max} - t)/t_{\max} + a_1 \tag{9}$$

$$a = (a_1 - a_0) \cdot (t/t_{\max})^2 + (a_0 - a_1) \cdot (2 \cdot t/t_{\max}) + a_1 \pm \tag{10}$$

$$a = a_0 + (a_1 - a_0) \cdot d_i \tag{11}$$

$$d_i = \frac{\|x_i - x_{best}\|}{d_{\max}} \tag{12}$$

where $t_{\max}$ expresses the total number of iterations, t is the current number of iterations, and $d_{\max}$ is the maximum distance between the optimal nest and all other nests. $a_0 < a_1$, a_0 is the initial value of step size.

In Q-Learning algorithm, the agent receives feedback, which is called reward, for each action. When the state is set to s and the action is set to a, a set of actions is set to $H = \{a_1, a_2, \ldots a_n\}$, the agent has n actions to choose from each state, and the maximum reward of discount for the agent is:

$$Q(s,a) = r(s_t, a_t) + \gamma \cdot \max_{a'} Q(s', a') \tag{13}$$

where $r(s,a)$ is the immediate benefits for state s. $\max_{a'} Q(s', a')$ is the maximum return value that the agent select different actions at the next state s'. a' is the action which is selected at the next state s'. γ is the discount factor. The benefits that the agent selecting action a receives is:

$$Q(a) = r(a) + \gamma \cdot Q(a^{(1)}) + \gamma^2 \cdot Q(a^{(2)}) + \ldots + \gamma^m \cdot Q(a^{(m)}) \tag{14}$$

where m represents the number of steps forward, $a, a^{(i)} \in A$, $1 \le i \le m$. When $\gamma = 0$, Q is reduced to one step forward. When γ is close to 1, the lagging benefits of optimal action increase gradually. $r(a)$ is the immediate benefit that the agent selects action a, which expresses that individuals have evolved once, and new individuals use $(a^{(1)})$ to generate new individuals again. At this time, the benefit is recorded as $Q(a^{(1)})$. By analogy, after m evolution, a new individual is generated by using $(a^{(m)})$, and the corresponding benefit is recorded as $Q(a^{(m)})$.

n offspring will be generated after each evolution. These offspring are evolved again by adopting n strategies. n^m offspring will be produced after m evolutions. Boltzmann distribution is used to calculate the probability of new individuals retained. Boltzmann distribution can be defined by Equation (15):

$$p(a_i) = e^{\frac{r(a_i)}{T}} / \sum_i^n e^{\frac{r(a_i)}{T}} \tag{15}$$

where $r(a_i)$ indicates the immediate benefits of the ith step strategy and T represents the temperature.

The step size control strategy corresponding to the maximum probability is selected. The results of each generation are simplified by Boltzmann distribution. $f_p(a)$ is defined as the fitness function corresponding to parent individual in the population and $f_o(a)$ is the fitness function corresponding to the individual after adopting the parameter selection strategy. Substituting $r(a) = f_p(a) - f_o(a)$ into Equation (13) Equation (16) is obtained.

$$Q(a) = f_p(a) - (1-\gamma) \cdot f_o(a^{(1)}) - \gamma \cdot (1-\gamma) \cdot f_o(a^{(2)}) - \ldots - \gamma^m \cdot f_o(a^{(m)}) \tag{16}$$

where $\forall m$, $\lim_{\to 1}^{m}(1-) = 0$ $\lim_{\to 1}^{m} = 1$; according to Equation (17), it can be concluded that $\lim_{\to 1} Q(a) = f_p(a) - f_o(a^{(m)})$, $a' = \text{argmax}_a \in A \lim \to 1 Q(a) = \text{argmax}_a \in A(f_p(a) - f_o(a^{(m)}))$. The step size control strategy model with Q-Learning is described in Algorithm 2 and Figure 1.

Algorithm 2 Step size with *Q*-Learning.

(1) Each individual is expressed as (x, σ), and the number of learning steps M is set;
(2) Generate three new offspring for each individual by using the given step size control strategy (Linear decreasing strategy, non-linear decreasing strategy, adaptively step-size dynamic adjustment strategy), and set $t = 1$;
(3) Do while $t < m$
Each individual generates three offspring by using the given step size control strategy, as shown in Equations (9)–(12).
Calculate the probability of the newly generated offspring by using the Boltzmann distribution, and an individual is selected according to the probability.
$t = t + 1$;
(4) Calculate the corresponding Q value of each retained individual according to the three-step selection strategy. The step size corresponding to the step control strategy is retained when Q is maximized, the corresponding offspring are selected, and other offspring will be discarded.

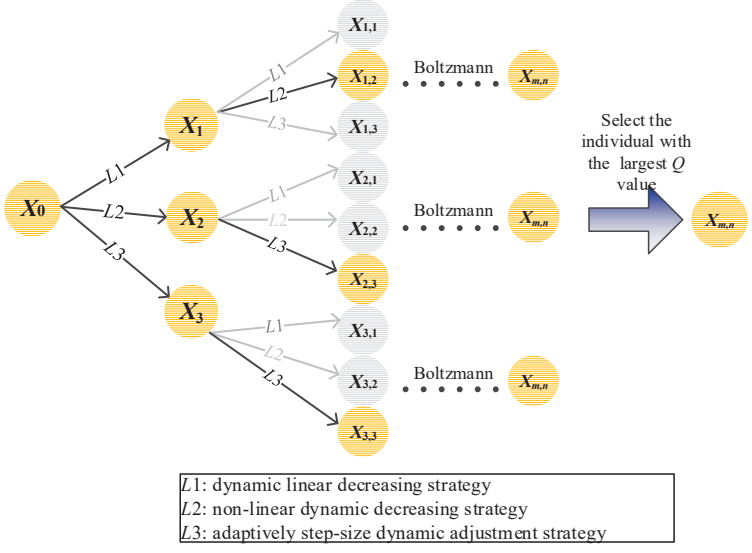


Figure 1. Step selection model with *Q*-Learning.

4.3. Genetic Operation

4.3.1. Crossover Process

As we know, two of the most important operators are the crossover operator and mutation operator for genetic operation [95], which have a great influence on the behavior and performance of genetic operation. Therefore, these operations are introduced into the DMQL-CS algorithm. In crossover process, a parameter C_r is defined as the probability of crossover and chromosomes are divided into pairs. We introduce the specifically designed crossover operation into the problem of logistics distribution center location in this paper, and apply it to a pair of chromosomes G_1 and G_2, as illustrated in Figure 2. First, some genes are randomly selected in chromosome G_1, as those pointed to by a red arrow in the illustration. Then, these genes are found in chromosome G_2, as pointed to by a green arrow. If the same gene is not found in G_2, two genes are randomly selected as the crossover point. Generate one child as the combination of red-pointed genes in G_1 and the rest of blue genes in G_2, and generate another child as the combination of green-pointed genes in G_2 and the rest of blue genes in G_1. Finally, the optimal is found as an arc between any two nodes by using enumeration method,

which keep the child to obtain the lowest objective value, and obtain chromosomes R_1 and R_2. Two chromosomes are selected from the parents and children with the smallest objective values to replace the parents.

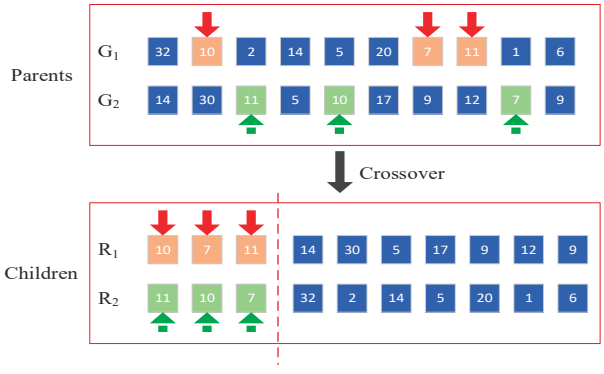


Figure 2. Crossover process of chromosomes.

At the same time, the crossover rate (C_r) is a critical factor for how the crossover operator behaves, which determines the performance of the algorithm to some extent. To improve the search ability of the algorithm, a substantial number of strategies have been designed to adjust the crossover rate. In this work, a self-adaptive scheme was used to adjust the crossover rate, which can be calculated as shown below.

$$C_r = 1/[1 + \exp(K_1)\mho] \tag{17}$$

where $= f_{avg} - f_{\max}$, f_{avg} is the average fitness, $f_{\max}$ is the max fitness, and K_1 is the scale factor between 0 and 1, $K_1 = 0.02$.

4.3.2. Mutation Process

A parameter C_m is defined as the mutation probability. The number r is randomly generated in the interval [0, 1]. If $r < C_m$, the ith chromosome G_1 is selected to perform the mutation operation and this process is repeated at each iteration. For illustration, we continue to use the problem of logistics distribution center location with 40 cities and 10 distribution centers. Two genes located on chromosome G_1 are randomly selected and their positions swapped to obtain a possible child. Then, the optimal is found as an arc between any two nodes by using enumeration method, which keeps the child obtaining the lowest objective value. Finally, we get chromosome R_1, as shown in Figure 3. If R_1 has a smaller objective value than G_1, G_1 is replaced with R_1, else G_1 is retained. A new generation of population is generated after the evaluation, crossover, and mutation operations.

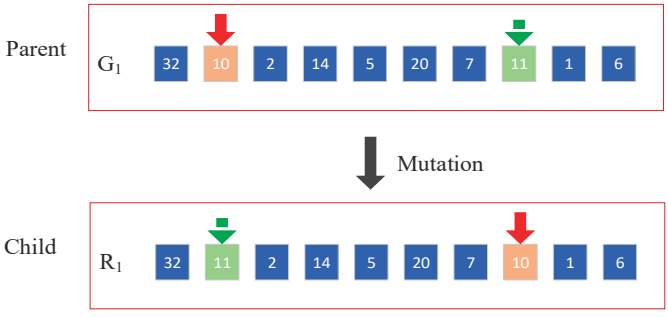


Figure 3. Mutation process of chromosomes.

4.3.3. Cuckoo Search Algorithm with *Q*-Learning Model and Genetic Operator

Introducing *Q*-Learning into CS algorithm helps to learn the optimal step size strategy and transform. Crossover and mutation strategies enable the nest to approach the historical optimal nest quickly, which can speed up the global convergence rate. The structure of the genetic operator cuckoo search algorithm with *Q*-Learning model (DMQL-CS) is described in Algorithm 3.

Algorithm 3 DMQL-CS Algorithm.

Input: Population size, *NP;* Maximum number of function evaluations, *MAX_FES*, *LP*
(1) Randomly initialize position of *NP* nest, *FES* = *NP*;
(2) Calculate the fitness value of each initial solution;
(3) **while** (stopping criterion is not meet do)
(4) Select the best step size control strategy according to Algorithm 2;
(5) Generate new solution x_i^{t+1} with the new step size by Lévy flights;
(6) Randomly choose a candidate solution x_i^t;
(7) **if** $f(x_i^t) > f(x_i^{t+1})$
(8) Replace x_i^t with new solution x_i^{t+1};
(9) **end if**
(10) Generate new solution x_i^{t+1} by using crossover operator and mutation operator;
(11) Throw out a fraction (p_a) of worst nests, generate solution k_i^{t+1} using Equation (3);
(12) **if** $f(x_i^t) > f(x_i^{t+1})$
(13) Replace x_i^t with new solution x_i^{t+1};
(14) **end if**
(15) Rank the solution and find the current best.
(16) **end while**

4.3.4. Analysis of Algorithm Complexity

To show the convergence effect of the algorithm, typical function Rastrigrin was selected to analyze the convergent process of DMQL-CS algorithm. Figure 4 shows the location distribution of cuckoo individuals in the search area with a population size of 10. Figure 4a describes the individual distribution at the first generation, Figure 4b describes the individual distribution at the 30th generation, Figure 4c describes the individual distribution at the 50th generation, and Figure 4d describes the individual distribution at the 80th generation. In Figure 4, it can be seen that the activity area of individuals keeps changing and gradually draws closer to the optimal solution during the evolution of the algorithm. It is worth noting that algorithm converged at the 80th generation, which indicates that *Q*-learning and genetic operation expand activity area of the population and improve the convergence performance of the DMQL-CS algorithm.

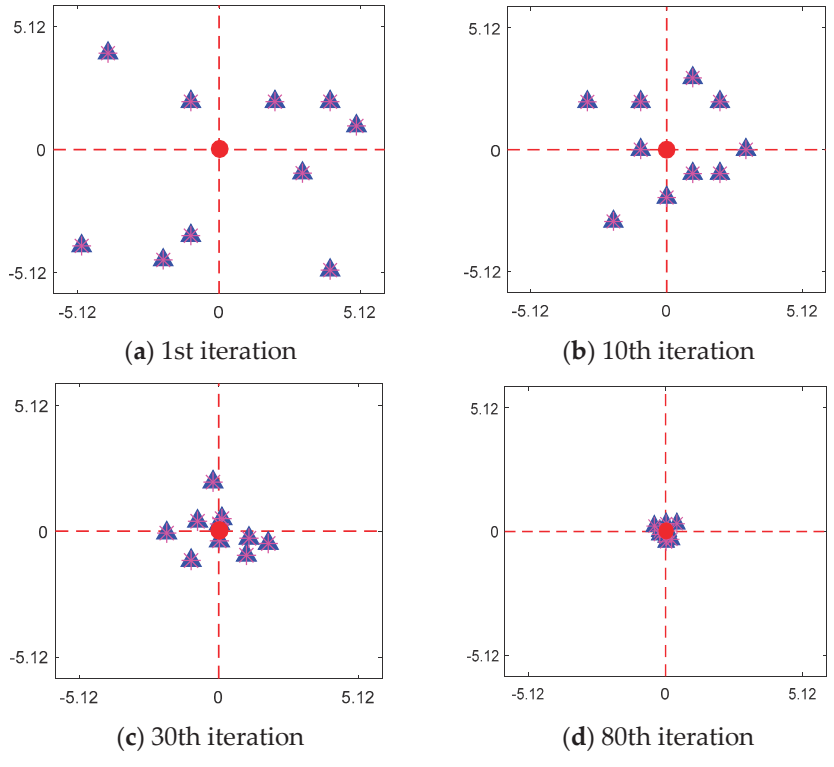


Figure 4. Analysis of algorithm complexity.

5. Results

5.1. Optimization of Functions and Parameter Settings

In this section, to check and verify the efficacy DMQL-CS algorithm, it is thoroughly investigated through benchmark evaluations from various respects. We tested our algorithms on two function groups: Group *A* and Group *B*. Group *A* contains fourteen different global optimization problems, as shown in Table 1. Group *B* is the CEC 2013 test suite including 28 benchmark functions. To make a fair comparison, all experiments were carried out on a P4 Dual-core platform with a 1.75 GHz processor and 4 GB memory, running under the Windows 7.0 operating system. The algorithms were written in MATLAB R2017a. The following were set: maximum number of evaluation $MAX\_FES = NP \times 10^5$, population size $NP = 30$, run time $T = 30$, and probability of foreign eggs $p_a = 0.25$.

Table 1. Brief description of fifteen functions.

Type	Function	Name	Search Range	Acceptable Accuracy	Global Optimum
Unimodal	F1	Sphere	[−100, 100]	1×10^{-8}	0
	F2	Rosenbrock	[−30, 30]	1×10^{-8}	0
	F3	Step	[−100, 100]	1×10^{-8}	0
	F4	Schwefel2.22	[−10, 10]	1×10^{-8}	0
Multimodal Shifted multimodal	F5	Ackley	[−32, 32]	1×10^{-8}	0
	F6	Rastrigin	[−5.12, 5.12]	10	0
	F7	Griewank	[−600, 600]	0.05	0
	F8	Generalized Penalized1	[−50, 50]	1×10^{-8}	0
	F9	Generalized Penalized2	[−50, 50]	1×10^{-8}	0
	F10	Shifted Schwefels Problem 1.2	[−100, 100]	1×10^{-8}	−450
	F11	Shifted Rotated High Conditioned Elliptic Function	[−100, 100]	1×10^{-8}	−450
	F12	Shifted Rosenbrock	[−100, 100]	2	390
	F13	Shifted Rotated Ackleys	[−32, 32]	2	−140
	F14	Shifted Griewanks	[−600, 600]	0.2	0
	F15	Shifted Rotated Rastrigin	[−5.12, 5.12]	10	−330

5.2. Comparison with Other CS Variants and Rank Based Analysis

We compared the performance of DMQL-CS with four improved CS variants: CCS [68], GCS [96], CSPSO [97], and OLCS [71]. CCS is a modified Chaos enhanced Cuckoo search algorithm. GCS introduces Gaussian disturbance into the CS algorithm. CSPSO is a kind of algorithm combining CS with PSO. A new search strategy based on orthogonal learning strategy is used in OLCS to enhance the exploitation ability of CS algorithm. The parameter configurations of these algorithms are shown in Table 2 according to corresponding references. Fifteen benchmark functions are shown in Tables 3–6 at $D = 30$ and $D = 50$. All optimization algorithms were tested by using the same parameter settings: population size $NP = 30$, $MAX\_FES = 100{,}000 \times D$, probability switching parameter $p_a = 0.25$, and run time $T = 30$.

As shown in Table 3, the DMQL-CS find global optima 0.00 on the four benchmark functions F1, F6, F7, and F14 when $D = 30$. For unimodal functions F1–F5, the DMQL-CS algorithm achieves higher accuracy than other CS variants on functions F2, F4, and F5. DMQL-CS is only inferior to OLCS on F2. For multimodal problems F6–F11, DMQL-CS algorithm shows higher performance than the other CS variants on functions F6, F7, F8, and F11. For F10, the same solution is found by the four algorithms (CCS, GCS, CSPSO, and OLCS). For the shifted unimodal functions F13–F15, DMQL-CS is significantly better than CCS, GCS, OLCS, and CSPSO on F13, F14 and F15. For F12, CCS performs the best.

Table 2. The personal parameters of different algorithms.

Algorithms	Parameter Configurations
CCS [68]	$p_a = 0.2$, $a = 0.5$, $b = 0.2$, $x_i = (0, 1)$
GCS [96]	$a = 1/3$, $p_a = 0.25$
CSPSO [97]	$p_a = 0.25$, $a = 0.1$, $W = 0.9$~0.4, $c_1 = c_2 = 2.0$
OLCS [71]	$p_a = 0.2$, $a = 0.5$, $K = 9$, $Q = 3$
DMQL-CS	$p_a = 0.25$, $M = 3$, $\gamma = 0.5$

Table 3. The optimization results obtained by CCS, GCS, CSPSO, OLCS, and DMQL-CS at $D = 30$.

Func	CCS	GCS	CSPSO	OLCS	DMQL-CS
F1	$3.21 \times 10^{-12} \pm 2.09 \times 10^{-12}$	$4.34 \times 10^{-10} \pm 3.23 \times 10^{-11}$	$4.77 \times 10^{-45} \pm 3.65 \times 10^{-44}$	$2.89 \times 10^{-106} \pm 1.43 \times 10^{-105}$	$\mathbf{0.00 \pm 0.00}$
F2	$3.96 \times 10^{-5} \pm 8.01 \times 10^{-5}$	$1.54 \times 10^{-1} \pm 1.82 \times 10^{-1}$	$1.66 \times 10^{-1} \pm 2.95 \times 10^{0}$	$1.45 \times 10^{-7} \pm 4.01 \times 10^{-7}$	$\mathbf{0.76 \times 10^{-7} \pm 5.12 \times 10^{-7}}$
F3	$4.12 \times 10^{0} \pm 3.11 \times 10^{0}$	$7.09 \times 10^{0} \pm 2.13 \times 10^{0}$	$7.12 \times 10^{0} \pm 3.31 \times 10^{0}$	$\mathbf{0.00 \pm 0.00}$	$4.88 \times 10^{-2} \pm 5.19 \times 10^{-1}$
F4	$5.56 \times 10^{-33} \pm 3.21 \times 10^{-32}$	$4.11 \times 10^{-24} \pm 5.01 \times 10^{-23}$	$\mathbf{2.76 \times 10^{-76} \pm 4.43 \times 10^{-76}}$	$4.09 \times 10^{-34} \pm 3.88 \times 10^{-34}$	$8.88 \times 10^{-35} \pm 5.78 \times 10^{-34}$
F5	$4.67 \times 10^{-5} \pm 3.21 \times 10^{-6}$	$4.11 \times 10^{-14} \pm 5.01 \times 10^{-13}$	$2.76 \times 10^{-2} \pm 4.43 \times 10^{-2}$	$7.21 \times 10^{-15} \pm 0.00$	$\mathbf{1.01 \times 10^{-15} \pm 2.87 \times 10^{-14}}$
F6	$6.22 \times 10^{-7} \pm 1.12 \times 10^{-5}$	$3.13 \times 10^{-7} \pm 2.98 \times 10^{-6}$	$3.87 \times 10^{1} \pm 2.01 \times 10^{1}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
F7	$3.13 \times 10^{-10} \pm 1.11 \times 10^{-10}$	$2.87 \times 10^{-11} \pm 2.12 \times 10^{-10}$	$5.77 \times 10^{-6} \pm 3.03 \times 10^{-6}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
F8	$4.90 \times 10^{-7} \pm 2.77 \times 10^{-7}$	$3.96 \times 10^{-7} \pm 3.31 \times 10^{-5}$	$1.39 \times 10^{-6} \pm 1.17 \times 10^{-5}$	$\mathbf{1.88 \times 10^{-8} \pm 4.09 \times 10^{-8}}$	$3.38 \times 10^{-7} \pm 2.99 \times 10^{-7}$
F9	$2.22 \times 10^{-23} \pm 1.05 \times 10^{-22}$	$3.04 \times 10^{-22} \pm 1.99 \times 10^{-22}$	$4.67 \times 10^{-4} \pm 2.89 \times 10^{-6}$	$\mathbf{4.39 \times 10^{-29} \pm 6.50 \times 10^{-26}}$	$2.45 \times 10^{-22} \pm 6.89 \times 10^{-22}$
F10	$6.01 \times 10^{-15} \pm 3.77 \times 10^{-16}$	$3.66 \times 10^{-15} \pm 2.19 \times 10^{-16}$	$\mathbf{3.21 \times 10^{-16} \pm 5.33 \times 10^{-16}}$	$3.21 \times 10^{-15} \pm 3.17 \times 10^{-11}$	$9.55 \times 10^{-15} \pm 7.09 \times 10^{-13}$
F11	$2.76 \times 10^{9} \pm 5.77 \times 10^{9}$	$2.81 \times 10^{9} \pm 3.06 \times 10^{9}$	$2.28 \times 10^{9} \pm 9.02 \times 10^{8}$	$5.63 \times 10^{6} \pm 2.22 \times 10^{6}$	$\mathbf{5.11 \times 10^{6} \pm 3.90 \times 10^{6}}$
F12	$\mathbf{1.23 \times 10^{1} \pm 2.77 \times 10^{0}}$	$1.42 \times 10^{1} \pm 2.93 \times 10^{0}$	$5.23 \times 10^{1} \pm 2.91 \times 10^{+1}$	$2.65 \times 10^{1} \pm 4.23 \times 10^{0}$	$4.21 \times 10^{1} \pm 1.09 \times 10^{1}$
F13	$1.90 \times 10^{3} \pm 3.97 \times 10^{3}$	$5.88 \times 10^{3} \pm 3.08 \times 10^{3}$	$4.34 \times 10^{4} \pm 1.88 \times 10^{3}$	$4.70 \times 10^{3} \pm 2.26 \times 10^{3}$	$\mathbf{2.06 \times 10^{2} \pm 3.77 \times 10^{1}}$
F14	$2.71 \times 10^{-1} \pm 2.09 \times 10^{0}$	$4.01 \times 10^{-1} \pm 7.00 \times 10^{0}$	$1.37 \times 10^{-2} \pm 8.01 \times 10^{-2}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
F15	$5.90 \times 10^{1} \pm 3.78 \times 10^{0}$	$7.88 \times 10^{1} \pm 2.89 \times 10^{0}$	$0.98 \times 10^{2} \pm 3.56 \times 10^{1}$	$3.65 \times 10^{1} \pm 4.11$	$\mathbf{2.87 \times 10^{1} \pm 4.77 \times 10^{1}}$

Table 4. The ranking of different strategies according to the Friedman test.

	CCS	GCS	CSPSO	OLCS	MP-QL-CS
Friedman rank	3.18	3.82	4.31	2.53	2.44
Final rank	3	4	5	2	1

Table 5. The optimization results obtained by CCS, GCS, CSPSO, OLCS, and DMQL-CS at $D = 50$.

Func	CCS	GCS	CSPSO	OLCS	MP-QL-CS
F1	$3.76 \times 10^{-6} \pm 2.21 \times 10^{-6}$	$2.78 \times 10^{-8} \pm 5.67 \times 10^{-9}$	$3.99 \times 10^{-19} \pm 6.43 \times 10^{-18}$	$4.45 \times 10^{-29} \pm 6.33 \times 10^{-28}$	$\mathbf{6.55 \times 10^{-30} \pm 2.90 \times 10^{-28}}$
F2	$3.88 \times 10^{1} \pm 3.09 \times 10^{1}$	$2.89 \times 10^{1} \pm 1.22 \times 10^{2}$	$5.98 \times 10^{-1} \pm 2.99 \times 10^{-1}$	$3.10 \times 10^{1} \pm 2.90 \times 10^{1}$	$\mathbf{1.99 \times 10^{-1} \pm 4.56 \times 10^{-1}}$
F3	$3.78 \times 10^{1} \pm 2.66 \times 10^{0}$	$3.67 \times 10^{1} \pm 4.52 \times 10^{0}$	$5.34 \times 10^{0} \pm 2.11 \times 10^{0}$	**0.00± 0.00**	$4.77 \times 10^{-2} \pm 3.21 \times 10^{-12}$
F4	$4.02 \times 10^{-2} \pm 1.55 \times 10^{-2}$	$3.78 \times 10^{-2} \pm 2.90 \times 10^{-2}$	$3.88 \times 10^{-4} \pm 1.89 \times 10^{-4}$	$4.65 \times 10^{-5} \pm 4.09 \times 10^{-5}$-	$\mathbf{4.90 \times 10^{-7} \pm 2.11 \times 10^{-8}}$
F5	$1.89 \times 10^{-2} \pm 2.87 \times 10^{-2}$	$5.97 \times 10^{-7} \pm 5.22 \times 10^{-7}$	$4.77 \times 10^{-2} \pm 4.44 \times 10^{-1}$	$5.09 \times 10^{-12} \pm 4.89 \times 10^{-14}$	$\mathbf{2.99 \times 10^{-12} \pm 3.09 \times 10^{-14}}$
F6	$5.34 \times 10^{-1} \pm 3.87 \times 10^{-1}$	$6.44 \times 10^{-6} \pm 3.72 \times 10^{-6}$	$9.28 \times 10^{3} \pm 4.73 \times 10^{3}$	**0.00 ± 0.00**	$3.98 \times 10^{-2} \pm 2.22 \times 10^{-1}$
F7	$3.12 \times 10^{-2} \pm 4.78 \times 10^{-2}$	$4.33 \times 10^{-2} \pm 9.21 \times 10^{-2}$	$6.34 \times 10^{-2} \pm 3.18 \times 10^{-2}$	**0.00 ± 0.00**	**0.00 ± 0.00**
F8	$6.67 \times 10^{-5} \pm 1.90 \times 10^{-5}$	$5.78 \times 10^{-7} \pm 3.77 \times 10^{-7}$	$8.90 \times 10^{-7} \pm 2.30 \times 10^{-7}$	$\mathbf{3.77 \times 10^{-8} \pm 7.56 \times 10^{-8}}$	$1.77 \times 10^{-4} \pm 2.12 \times 10^{-4}$
F9	$5.78 \times 10^{-3} \pm 0.55 \times 10^{-3}$	$7.78 \times 10^{-20} \pm 6.23 \times 10^{-20}$	$3.66 \times 10^{-1} \pm 3.41 \times 10^{-1}$	$\mathbf{4.67 \times 10^{-25} \pm 1.23 \times 10^{-26}}$	$3.90 \times 10^{-10} \pm 3.66 \times 10^{-9}$
F10	$5.78 \times 10^{-10} \pm 5.55 \times 10^{-10}$	$3.99 \times 10^{-10} \pm 2.98 \times 10^{-10}$	$7.90 \times 10^{-10} \pm 8.11 \times 10^{-10}$	$7.34 \times 10^{-10} \pm 5.45 \times 10^{-9}$	$\mathbf{2.78 \times 10^{-10} \pm 1.34 \times 10^{-6}}$
F11	$3.66 \times 10^{12} \pm 3.89 \times 10^{12}$	$2.89 \times 10^{12} \pm 5.78 \times 10^{12}$	$\mathbf{2.90 \times 10^{7} \pm 3.11 \times 10^{7}}$	$5.89 \times 10^{8} \pm 9.90 \times 10^{8}$	$4.89 \times 10^{11} \pm 3.67 \times 10^{10}$
F12	$4.89 \times 10^{3} \pm 3.78 \times 10^{3}$	$2.90 \times 10^{3} \pm 2.22 \times 10^{3}$	$5.98 \times 10^{3} \pm 2.09 \times 10^{3}$	$\mathbf{6.99 \times 10^{2} \pm 3.90 \times 10^{1}}$	$2.97 \times 10^{3} \pm 1.86 \times 10^{3}$
F13	$4.89 \times 10^{5} \pm 2.17 \times 10^{5}$	$5.89 \times 10^{4} \pm 1.12 \times 10^{4}$	$5.33 \times 10^{5} \pm 4.56 \times 10^{5}$	$5.02 \times 10^{4} \pm 2.09 \times 10^{4}$	$\mathbf{1.09 \times 10^{3} \pm 3.89 \times 10^{3}}$
F14	$6.98 \times 10^{1} \pm 1.11 \times 10^{2}$	$5.56 \times 10^{1} \pm 2.98 \times 10^{2}$	$5.89 \times 10^{2} \pm 2.21 \times 10^{2}$	**0.00 ± 0.00**	$2.90 \times 10^{1} \pm 3.76 \times 10^{1}$
F15	$6.25 \times 10^{2} \pm 3.33 \times 10^{2}$	$4.28 \times 10^{2} \pm 1.77 \times 10^{2}$	$3.45 \times 10^{3} \pm 2.76 \times 10^{3}$	$8.89 \times 10^{2} \pm 4.11 \times 10^{2}$	$\mathbf{2.22 \times 10^{2} \pm 1.78 \times 10^{2}}$

Table 6. The ranking of different strategies according to the Friedman test.

	CCS	GCS	CSPSO	OLCS	MP-QL-CS
Friedman rank	4.19	3.62	4.15	2.41	2.46
Final rank	5	3	4	1	2

DMQL-CS still has outstanding optimization performance when $D = 50$, as shown in Table 5. From the results, it is apparent that the convergence precision of other algorithms drops rapidly, while the DMQL-CS algorithm achieves better performance than other CS variants on most functions. DMQL-CS and OLCS achieve the global optimum on function F7. DMQL-CS cannot get the minimum; even then, it is not inferior to other algorithms on F4, F5, F10, F12, F13, and F15. In addition, the DMQL-CS demonstrates a remarkable accuracy on benchmark F1 and F2. Comparing with the optimization results, we can conclude that the DMQL-CS optimization algorithm explored a larger search space than other CS variants. Moreover, it is important to point out that, regardless of the problem's dimensionality, the DMQL-CS converges to the better solution on the shifted multimodal functions F13, F14 and F15. Therefore, these statistical tests confirmed that DMQL-CS algorithm with Q-Learning step size and genetic operators has a better overall performance than all other tested competitors. For a clearer observation that DMQL-CS performs best, Table 4 shows the ranking of the strategies in Table 3 according to the Friedman test. We can see that DMQL-CS obtains the best rank, OLCS ranks second, followed by CCS, GCS, and CSPSO. Table 6 shows the ranking of the five strategies according to the Friedman test. OLCS obtains the best rank, DMQL-CS ranks second, followed by GCS, CSPSO, and CCS.

To further demonstrate the convergence of DMQL-CS, the median convergence properties of five algorithms are illustrated in Figure 5. There is no obvious "evolution stagnation" for all algorithms. For the same population size and number of generations, the optimization performance of the four algorithms declines rapidly. However, DMQL-CS can get better convergence curve than CCS, GCS, CSPSO, and OLCS on F1–F2, F5–F6, F12, and F14. In Figure 5, DMQL-CS algorithm converged to the specified error threshold on function F1, which suggests that DMQL-CS algorithm has a faster convergence rate for the specified error threshold. Generally speaking, when M is too small, useful step size information will not be learned. When M is too large, the speed of Q-Learning will be slowed down. When the value of M is 3 or 5, the convergence performance of DMQL-CS can be improved for the ill-condition function F2, complex multimodal functions F5–F6, and Shifted multimodal functions F12 and F14. It is worth mentioning that the accuracy of OLCS is similar to that of DMQL-CS, but the convergence speed of DMQL-CS is much faster than that of OLCS. For multimodal function, all algorithms converge to the specified error threshold with the same number of successes. However, DMQL-CS has good reliability, stability, and faster convergence rate on functions F5–F6. For function F14, DMQL-CS algorithms can find the global optimum with 50,000 FES. As mentioned above, it can be clearly observed that DMQL-CS provided better performance than the four other CS versions, and achieves a promising solution on most test functions.

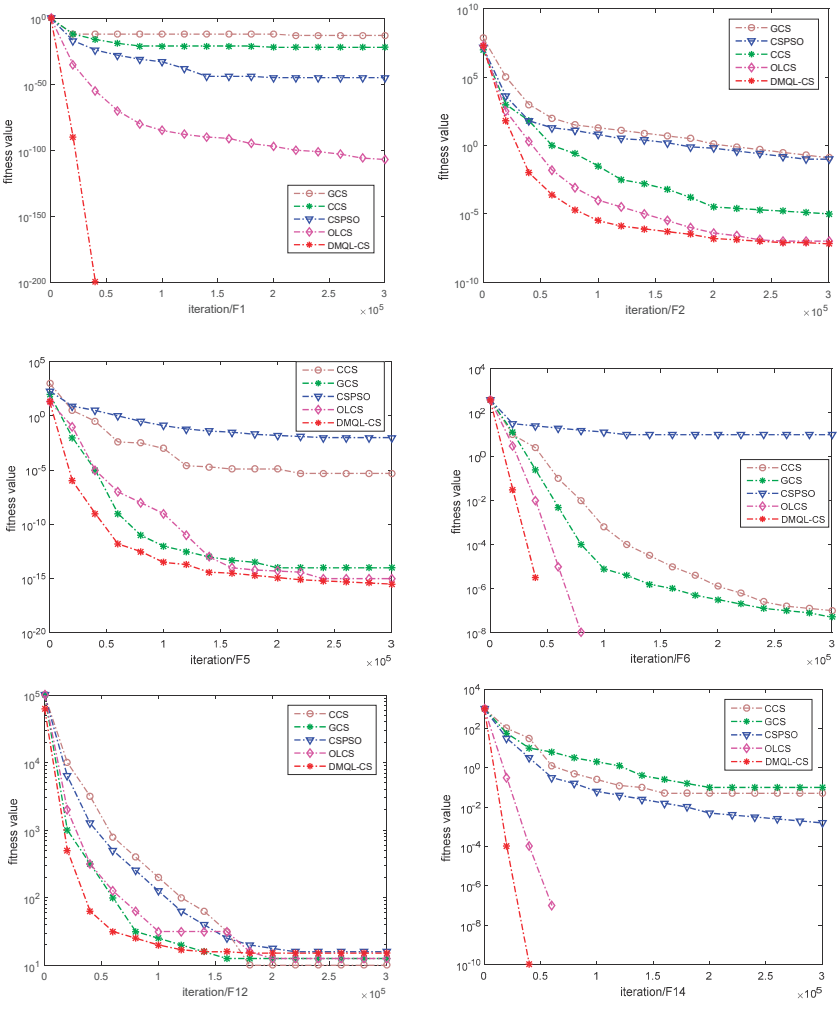


Figure 5. Convergence curves of different algorithms on test functions when $D = 30$.

A series of comparisons proved the high efficiency of DMQL-CS. The performance ranking of the multiple algorithms of the test suite is listed in Tables 7–9. The average rankings of the five CS variants for functions F1–F8 are shown in Table 7 ($D = 30$ and $D = 50$). The average rankings of the five CS variants for functions F9–F15 are shown in Table 8 ($D = 30$ and $D = 50$). In competition ranking, if performances of algorithms are the same, they received the same rank. It can be seen in Tables 7–9 that the average ranking value of DMQL-CS on $D = 30$ is smaller than that of CCS, GCS, OLCS, and CSPSO. Therefore, the performance of DMQL-CS is better than the other CS variants. When $D = 50$, the results are similar to those when $D = 30$, with the average ranking value of DMQL-CS being smaller than those of CCS, GCS, OLCS, and CSPSO.

Table 7. Rank results of each algorithm on F1–F8 for $D = 30$ and $D = 50$.

Dim	Algorithm	F1	F2	F3	F4	F5	F6	F7	F8
30	**CCS**	4	3	3	3	4	4	4	4
	GCS	5	4	4	2	3	3	3	3
	CSPSO	3	5	5	4	5	5	5	5
	OLCS	2	1	1	5	2	1	1	2
	DMQL-CS	1	2	2	1	1	1	1	1
50	**CCS**	5	5	5	5	4	4	3	3
	GCS	4	3	4	4	3	2	4	2
	CSPSO	3	2	3	3	5	5	5	4
	OLCS	2	4	1	2	2	1	1	1
	DMQL-CS	1	1	2	1	1	3	1	5

Table 8. Rank results of each algorithm on F9-F15 for $D = 30$ and $D = 50$.

Dim	Algorithm	F9	F10	F11	F12	F13	F14	F15
30	**CCS**	3	4	4	1	2	4	4
	GCS	4	3	5	2	4	3	3
	CSPSO	5	1	3	5	5	2	5
	OLCS	1	2	2	3	3	1	2
	DMQL-CS	2	5	1	4	1	1	1
50	**CCS**	4	3	5	4	4	3	2
	GCS	2	2	4	2	3	4	4
	CSPSO	5	5	1	5	5	5	3
	OLCS	1	4	2	1	2	1	5
	DMQL-CS	3	1	3	3	1	2	1

Table 9. Total rank and final rank on F1–F15 for $D = 30$ and $D = 50$.

Dim	Rank	Algorithms				
		CCS	**GCS**	**CSPSO**	**OLCS**	**DMQL-CS**
30	Total rank	49	53	63	29	25
	Final rank	3	4	5	2	1
50	Total rank	60	47	55	30	29
	Final rank	5	3	4	2	1

In Table 9, DMQL-CS has the best total rank when $D = 30$ and $D = 50$, i.e., 25 and 29, which means that DMQL-CS has the best performance on most of the test functions compared with other algorithms. OLCS has the second-best total rank at $D = 30$ and $D = 50$, i.e., 29 and 30. Obviously, OLCS has better performance than the three other algorithms on high-dimension test functions. Similarly, in Table 9, the order can be clearly observed: DMQL-CS > OLCS > CCS > GCS > CSPSO at $D = 30$; and DMQL-CS > OLCS > GCS > CCS > CSPSO at $D = 50$. Based on the analysis of the above, DMQL-CS has the best performance among all the algorithms on both $D = 30$ and $D = 50$.

5.3. Statistical Analysis of Performance for the CEC 2013 Test Suite

In this section, the CEC 2013 test suite is selected to test the effectiveness of three different algorithms (jDE [98], SaDE [99], and CLPSO [100]). These algorithms can be seen as representatives of the state-of-the-art algorithms for comparison, and the parameter configurations of these algorithms were set according to the corresponding references, as listed in Table 10.

Table 10. The personal parameters of different algorithms.

Algorithms	Parameter Configurations
jDE [98]	$F = 0.5$, $CR = 0.9$
SaDE [99]	F~N(0.5, 0.3), $CR_0 = 0.5$, CR~N(CR_m, 0.1), $LP = 50$
CLPSO [100]	$W = 0.7298$, $c = 1.49618$, $m = 7$, $p_c = 0.05$~0.5
DMQL-CS	$p_a = 0.25$, $PL = 20$, $\eta = 0.015$

Table 11 summarizes the results of CEC 2013 test problems on 28 benchmark functions for 30-dimensional case. The rank was used to obtain the ranking of different algorithms on all problems, as shown in Table 12. This means that DMQL-CS gets the first rank and outperforms jDE, SaDE, and CLPSO. The results in Table 11 indicate that with 80% certainty DMQL-CS has statistically higher accuracy than the other algorithms. Note that DMQL-CS obtains the global optimal value 0.00 on F1 and F11. DMQL-CS is significantly better than the three other algorithms, especially on functions CEC 2013-F1, CEC 2013-F2, and CEC 2013-F4. About basic Multimodal Function and composition Functions (CEC 2013-F6–CEC 2013-F28), the ability of DMQL-CS to find the optimal solution is slightly better than that of CLPSO. For functions CEC 2013-F5, CEC 2013-F7, CEC 2013-F17, CEC 2013-F22, and CEC 2013-F25, the performance of DMQL-CS is slightly worse than the other algorithms. For the Unimodal problem CEC 2013-F3, jDE obtains the best solution 2.99×10^6. On Shift Rastrigin Function, SaDE and jDE can get better solutions of 1.10×10^1 and 1.06×10^{-4}, respectively. For CEC 2013-F25–CEC 2013-F26, DMQL-CS is obviously better than SaDE and jDE. CLPSO has the weakest ability to find the optimal solution for 28 functions. From the above results, it can be seen that DMQL-CS with Q-Learning and genetic operations has a better overall performance than all other tested competitors on the CEC 2013 test suite. Table 13 reports the rankings of the results between DMQL-CS and other algorithms. In Table 13, it can be seen that DMQL-CS performs the best among the four algorithms. DMQL-CS exhibits consistent ranks of the first in optimizing most of the functions. For a clearer observation that DMQL-CS performs best, Table 12 shows the ranking of the algorithms in according to the Friedman test. DMQL-CS obtains the best rank, jDE ranks second, followed by SaDE and CLPSO.

Table 11. The mean and standard deviation (STD) of CEC 2013 test suite with four algorithms.

Function	Mean Std	Algorithms			
		SaDE	jDE	CLPSO	DMQL-CS
CEC 2013-F1	Mean/Std	0.00/0.00	**0.00/0.00**	2.16×10^{-13}/**0.00**	**0.00/0.00**
CEC 2013-F2	Mean/Std	4.21×10^{5}/1.21×10^{5}	1.27×10^{5}/6.86×10^{5}	2.99×10^{7}/2.10×10^{6}	**1.12×10^{5}/4.88×10^{5}**
CEC 2013-F3	Mean/Std	2.98×10^{7}/2.99×10^{7}	**2.99×10^{6}/3.01×10^{6}**	3.16×10^{8}/2.92×10^{8}	3.78×10^{7}/5.87×10^{6}
CEC 2013-F4	Mean/Std	3.22×10^{3}/2.98×10^{3}	0.97×10^{1}/1.88×10^{1}	4.87×10^{4}/1.09×10^{3}	**8.09×10^{0}/7.90×10^{0}**
CEC 2013-F5	Mean/Std	**0.00/0.00**	1.19×10^{-13}/3.55×10^{-14}	3.54×10^{-11}/2.01×10^{-12}	4.56×10^{-14}/1.90×10^{-12}
CEC 2013-F6	Mean/Std	2.78×10^{1}/5.66×10^{1}	1.23×10^{1}/9.78×10^{0}	3.56×10^{1}/1.00×10^{1}	**5.62×10^{-2}/9.89×10^{0}**
CEC 2013-F7	Mean/Std	2.22×10^{1}/**1.38×10^{0}**	**2.12×10^{1}/1.38×10^{0}**	6.97×10^{1}/3.20×10^{1}	8.90×10^{1}/5.12×10^{0}
CEC 2013-F8	Mean/Std	2.11×10^{1}/6.45×10^{1}	**2.01×10^{0}**/6.11×10^{0}	2.09×10^{1}/3.73×10^{-2}	2.07×10^{1}/**1.78×10^{-2}**
CEC 2013-F9	Mean/Std	1.78×10^{1}/**2.33×10^{0}**	2.59×10^{1}/4.45×10^{0}	3.19×10^{1}/3.62×10^{0}	**1.01×10^{1}**/7.90×10^{0}
CEC 2013-F10	Mean/Std	2.73×10^{-1}/2.33×10^{-1}	4.37×10^{-2}/**4.73×10^{-2}**	3.99×10^{1}/2.01×10^{0}	**2.77×10^{-3}**/6.89×10^{-2}
CEC 2013-F11	Mean/Std	3.87×10^{-2}/3.64×10^{-1}	**0.00/0.00**	6.14×10^{1}/3.01×10^{1}	**0.00/0.00**
CEC 2013-F12	Mean/Std	4.19×10^{1}/2.65×10^{1}	5.56×10^{1}/**1.49×10^{1}**	1.23×10^{2}/2.11×10^{1}	**9.01×10^{0}**/4.67×10^{1}
CEC 2013-F13	Mean/Std	**1.10×10^{1}**/3.21×10^{2}	1.29×10^{1}/5.22×10^{1}	1.78×10^{2}/**3.91×10^{1}**	1.11×10^{2}/3.99×10^{2}
CEC 2013-F14	Mean/Std	7.34×10^{0}/2.22×10^{0}	**1.06×10^{-4}**/4.24×10^{-3}	5.77×10^{2}/4.79×10^{2}	1.89×10^{2}/**0.00**
CEC 2013-F15	Mean/Std	**4.90×10^{3}/5.67×10^{2}**	5.03×10^{3}/6.02×10^{2}	6.01×10^{3}/4.25×10^{2}	4.98×10^{3}/2.67×10^{3}
CEC 2013-F16	Mean/Std	2.24×10^{0}/5.12×10^{-1}	3.05×10^{0}/2.26×10^{-1}	3.22×10^{0}/2.21×10^{-1}	**1.18×10^{0}/2.05×10^{-1}**
CEC 2013-F17	Mean/Std	**6.12×10^{1}/1.16×10^{-2}**	6.13×10^{1}/3.65×10^{0}	7.23×10^{1}/5.51×10^{0}	5.89×10^{2}/4.24×10^{1}
CEC 2013-F18	Mean/Std	1.02×10^{2}/2.57×10^{1}	1.61×10^{2}/3.21×10^{1}	2.25×10^{1}/1.11×10^{1}	**1.78×10^{1}/1.12×10^{1}**
CEC 2013-F19	Mean/Std	4.91×10^{0}/6.18×10^{-1}	**1.05×10^{0}**/3.81×10^{-1}	3.18×10^{0}/**1.03×10^{-1}**	3.27×10^{0}/3.99×10^{-1}
CEC 2013-F20	Mean/Std	1.10×10^{1}/5.23×10^{0}	1.15×10^{1}/4.09×10^{0}	1.41×10^{1}/2.55×10^{0}	**1.02×10^{1}**/5.12×10^{0}
CEC 2013-F21	Mean/Std	2.80×10^{2}/3.55×10^{1}	2.65×10^{2}/1.29×10^{1}	3.12×10^{2}/4.11×10^{1}	**2.01×10^{2}**/4.88×10^{2}
CEC 2013-F22	Mean/Std	1.25×10^{3}/2.23×10^{2}	1.17×10^{3}/2.23×10^{2}	**7.35×10^{2}/1.01×10^{2}**	8.23×10^{2}/6.98×10^{2}
CEC 2013-F23	Mean/Std	4.83×10^{3}/0.21×10^{3}	4.90×10^{3}/**2.21×10^{2}**	6.23×10^{3}/3.83×10^{2}	**2.00×10^{3}**/3.23×10^{2}
CEC 2013-F24	Mean/Std	2.35×10^{2}/**2.56×10^{0}**	**2.21×10^{2}**/5.56×10^{1}	2.89×10^{2}/7.38×10^{0}	2.67×10^{2}/3.21×10^{0}
CEC 2013-F25	Mean/Std	2.45×10^{2}/1.28×10^{1}	2.66×10^{2}/**1.01×10^{0}**	2.80×10^{2}/6.33×10^{0}	**2.02×10^{2}**/3.98×10^{0}
CEC 2013-F26	Mean/Std	2.23×10^{2}/2.09×10^{3}	2.17×10^{2}/2.51×10^{1}	1.97×10^{2}/1.66×10^{-1}	**1.04×10^{2}/1.09×10^{2}**
CEC 2013-F27	Mean/Std	**5.48×10^{2}**/4.23×10^{1}	6.34×10^{2}/3.32×10^{2}	8.13×10^{2}/2.65×10^{2}	6.89×10^{2}/3.89×10^{2}
CEC 2013-F28	Mean/Std	3.05×10^{2}/**0.00**	3.05×10^{2}/**0.00**	3.03×10^{2}/2.98×10^{-3}	**3.02×10^{2}**/3.20×10^{2}

Table 12. The ranking of different strategies according to the Friedman test.

	SaDE	jDE	CLPSO	DMQL-CS
Friedman rank	3.34	2.95	4.26	2.59
Final rank	3	2	4	1

Table 13. Comparisons between DMQL-CS and other algorithms for CEC 2013 test suite.

Function	SaDE	jDE	CLPSO	DMQL-CS
	Rank	Rank	Rank	Rank
CEC 2013-*F*1	1 (≈/=)	1 (≈/=)	4 (-)	1
CEC 2013-*F*2	4 (-)	2 (-)	3 (-)	1
CEC 2013-*F*3	2 (-)	1 (+)	4 (-)	3
CEC 2013-*F*4	3 (-)	2 (-)	4 (-)	1
CEC 2013-*F*5	1 (+)	3 (+)	2 (+)	4
CEC 2013-*F*6	3 (-)	2 (-)	4 (-)	1
CEC 2013-*F*7	2 (+)	1 (+)	3 (+)	4
CEC 2013-*F*8	4 (-)	1 (+)	3 (-)	2
CEC 2013-*F*9	2 (-)	3 (-)	4 (-)	1
CEC 2013-*F*10	2 (-)	3 (-)	4 (-)	1
CEC 2013-*F*11	3 (-)	1 (≈/=)	4 (-)	1
CEC 2013-*F*12	2 (-)	3 (-)	4 (-)	1
CEC 2013-*F*13	1 (+)	3 (+)	4 (+)	2
CEC 2013-*F*14	2 (+)	1 (+)	4 (-)	3
CEC 2013-*F*15	1 (+)	3 (-)	4 (-)	2
CEC 2013-*F*16	2 (-)	3 (-)	4 (-)	1
CEC 2013-*F*17	1 (+)	2 (+)	3 (+)	4
CEC 2013-*F*18	2 (-)	3 (-)	4 (-)	1
CEC 2013-*F*19	4 (-)	1 (+)	2 (+)	3
CEC 2013-*F*20	2 (-)	3 (-)	4 (-)	1
CEC 2013-*F*21	3 (-)	2 (-)	4 (-)	1
CEC 2013-*F*22	3 (+)	2 (+)	1 (+)	4
CEC 2013-*F*23	2 (-)	3 (-)	4 (-)	1
CEC 2013-*F*24	2 (+)	1 (+)	4 (-)	3
CEC 2013-*F*25	1(+)	2 (+)	3 (+)	4
CEC 2013-*F*26	4 (-)	3 (-)	2 (-)	1
CEC 2013-*F*27	1 (+)	2 (+)	4 (-)	3
CEC 2013-*F*28	3 (-)	3 (-)	2 (-)	1
Rank_Sun	63	60	96	56
Rank_Final	3	2	4	1

5.4. Application in the Problem of Logistics Distribution Center Location

5.4.1. Problem Description

The location of logistics distribution center is an important link in the logistics system. The location of distribution center determines the efficiency of the entire logistics network system and the utilization of resources. The location selection model of logistics distribution center is a nonlinear model with more complex constraints and non-smooth characteristics, which can be attributed to HP-hard problem. The problem of logistics distribution center location can be described as: m cargo distribution centers are searched in n demand points, so that the distance between m searched distribution centers and other n cargo demand points is the shortest. At the same time, the following constraint conditions must be met: the supply of goods in the distribution center can meet the requirements of the cargo demand points; the goods required for a cargo demand point can only be provided by one distribution center; and the cost of transporting the goods to the distribution center is not considered. According

to the above assumptions, the mathematical model of the problem for logistics distribution center location can be described as:

$$\min(\cos t) = \sum_{i=1}^{m}\sum_{j=1}^{n}(need_j \cdot dist_{i,j} \cdot \mu_{i.j}) \tag{18}$$

$$s.t. \sum_{i=1}^{m} \mu_{i,j} = 1, i \in M, j \in N \tag{19}$$

$$\mu_{i,j} \le h_j, i \in M, j \in N \tag{20}$$

$$\sum_{t=1}^{m} h_i \le p, i \in M \tag{21}$$

$$h_j \in \{0,1\}, i \in M \tag{22}$$

$$\mu_{i,j} \in \{0,1\}, i \in M, j \in N \tag{23}$$

$$M = \{j \big| j = 1,2,\ldots,m\} \ N = \{j \big| j = 1,2,\ldots,n\} \tag{24}$$

where Equation (18) is the objective function, cost represents the transportation cost, m is the number of logistics distribution center, n determines the number of goods demand point, $nest_j$ is the demand quantity of demand point j, and $dist_{i,j}$ indicates the distance between distribution center i and goods demand point j. When $u_{i,j}$ is equal to 1, the goods of demand point j are distributed by distribution point i. Equations (19)–(24) are the constraints. Equation (19) defines that a demand point of goods can only be distributed by a distribution center. Equation (20) indicates that each demand point of goods must have a distribution center to distribute goods. Equation (21) represents the number of goods demand points for a distribution center. Equations (22)–(24) are the relevant definitions.

5.4.2. Analysis of Experimental Results

To verify the performance of the DMQL-CS algorithm in solving the problem of logistics distribution center location, 40 demand points were adopted. The geographical position coordinates and demands are shown in Table 14. To make a fair comparison, all experiments were carried out on a P4 Dual-core platform with a 1.75 GHz processor and 4 GB memory, running under the Windows 7.0 operating system. The algorithms were written by MATLAB R2017a. The maximum number of iterations, population size, and the times of running were set to 30,000, 15, and 30, respectively. The probability of foreign eggs being found was = 0.25.

Table 14. The geographical position coordinates and demands.

No	Coordinates		Demand	No	Coordinates		Demand	No	Coordinates		Demand	No	Coordinates		Demand
	x	y			x	y			x	y			x	y	
1	97	28	94	11	91	96	85	21	111	117	92	31	125	66	45
2	100	56	11	12	39	90	54	22	63	42	99	32	169	49	98
3	45	67	50	13	50	101	25	23	67	105	98	33	31	188	31
4	150	197	88	14	67	66	87	24	160	156	88	34	86	42	91
5	105	48	80	15	157	54	66	25	100	125	47	35	90	21	79
6	24	158	29	16	104	35	82	26	35	48	47	36	46	53	47
7	88	61	93	17	169	95	48	27	143	172	34	37	62	30	84
8	55	105	10	18	48	39	78	28	94	56	33	38	163	176	52
9	120	120	18	19	115	61	16	29	57	73	43	39	190	141	10
10	43	105	38	20	154	174	49	30	25	127	100	40	170	30	77

To further verify the efficiency of the DMQL-CS algorithm, the effectiveness of the proposed method was verified by comparing the standard cuckoo search algorithm (CS) [69], the improved cuckoo search algorithm (ICS) [101], a modified chaos-enhanced cuckoo search algorithm (CCS) [68], and the immune genetic algorithm (IGA) [64]. Figure 5 shows the average convergence curve and

optimal convergence curve of DMQL-CS algorithm for running 20, 30, and 50 times, respectively, in 40 cities and six distribution centers. The six optimal distribution center points and optimal routes found by DMQL-CS algorithm are also shown in Figure 6. Figure 7 shows the average convergence curve and optimal convergence curve of DMQL-CS algorithm running 20, 30, and 50 times, respectively, in 40 cities and 10 distribution centers. Table 15 shows distribution ranges for six distribution centers in 40 cities, and Table 16 shows distribution ranges for 10 distribution centers in 40 cities.

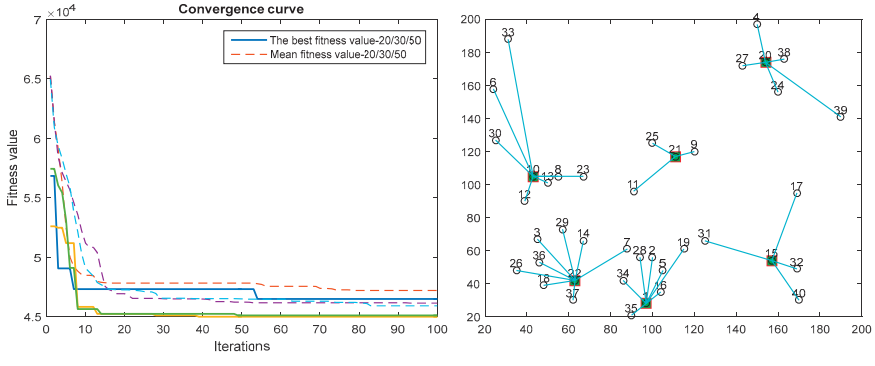


Figure 6. Convergence curves and optimal distribution centers scheme for the DMQL-CS algorithm in 6 distribution centers.

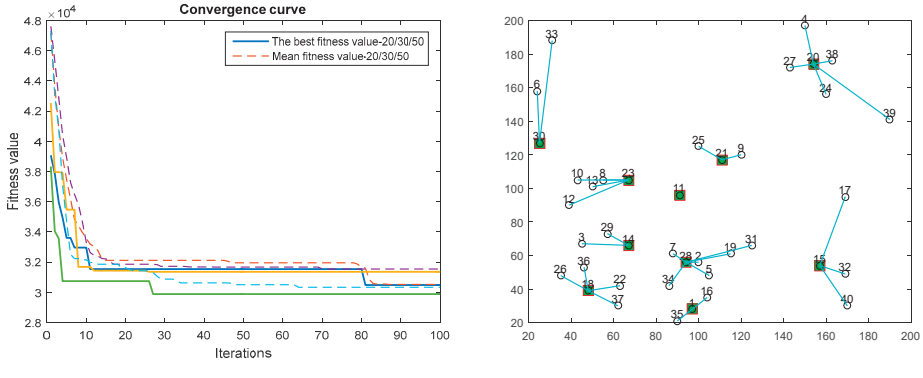


Figure 7. Convergence curves and optimal distribution centers scheme for the DMQL-CS algorithm in 10 distribution centers.

Table 15. The distribution scheme for six distribution centers in 40 cities.

Distribution Center	Distribution Scope
10	33, 6, 30, 12, 13, 8, 23
21	11, 25, 9
20	4, 27, 38, 24, 39
22	14, 29, 3, 36, 26, 18, 37, 7
1	28, 25, 16, 19, 34, 35
15	31, 17, 32, 40

Table 16. The distribution scheme for 10 distribution centers in 40 cities.

Distribution Center	Distribution Scope
30	6, 33
23	8,12, 13, 10
14	3, 29
18	26, 36, 22, 37
11	-
28	7, 34, 2, 19, 31, 5
21	25, 9
1	16, 35
20	4, 27, 38, 24, 39
15	17, 32, 40

For the first set of experiments, the DMQL-CS algorithm was run 20, 30, and 50 times independently in 40 cities for six distribution centers. As shown in Figure 6, the average convergence curve can converge at 30 iterations. It indicates that the fitness value decreases rapidly for the logistics distribution center location method based on DMQL-CS algorithm at early stage of the algorithm. The optimal distribution cost and average distribution cost obtained by the DMQL-CS algorithm are 4.5013×10^4 and 4.8060×10^4, respectively, which indicates that DMQL-CS has high solution accuracy for six distribution centers and reduces the cost of logistics distribution. The optimal distribution center points found in Figure 3 are: 10, 21, 20, 22, 1, and 15.

For the second set of experiments, the DMQL-CS algorithm was run 20, 30, and 50 times independently for 40 cities and 10 distribution centers. As shown in Figure 7, the average convergence curve can converge at 20 iterations. The optimal distribution cost and average distribution cost obtained by the DMQL-CS algorithm are 2.9811×10^4 and 3.0157×10^4, respectively, which indicates that DMQL-CS has high solution accuracy not only for six distribution centers but also for 10 distribution centers. The 10 optimal distribution centers and distribution addressing schemes are shown in Table 16 and Figure 7. The optimal distribution center points are: 30, 23, 14, 18, 11, 28, 21, 1, 20, and 15.

Due to limited space, only three comparison algorithms (CS [69], CCS [68], and IGA [64]) are listed in this paper. IGA algorithm introduced crossover and variation strategy into immune algorithm, which improves performance of the immune algorithm. In this experiment, the convergence curves and optimal distribution scheme diagrams of 6 and 10 distribution centers in 40 cities are shown, respectively. Figures 8 and 9 show the convergence curves and optimal distribution centers scheme for the IGA algorithm for 6 and 10 distribution centers. Figures 10 and 11 show the convergence curves and optimal distribution centers scheme for the CS algorithm for 6 and 10 distribution centers. Figures 12 and 13 show the convergence curves and optimal distribution centers scheme for CCS algorithm for 6 and 10 distribution centers. The six optimal distribution centers and distribution addressing schemes for these algorithms are shown in Table 17, while the 10 optimal distribution centers and distribution addressing schemes are shown in Table 18.

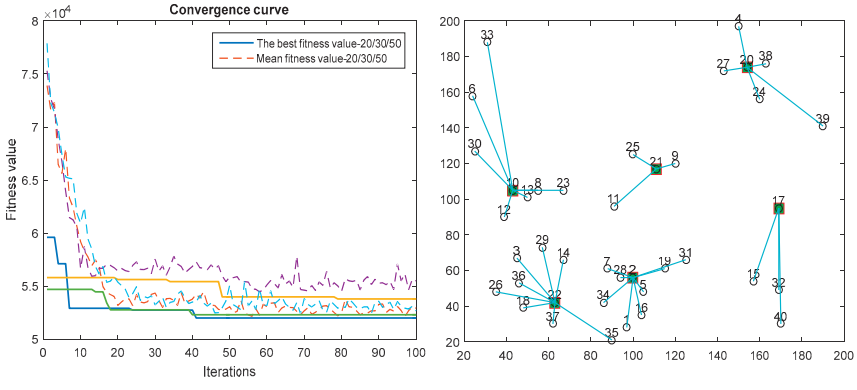


Figure 8. Convergence curves and optimal distribution centers scheme for the IGA algorithm for six distribution centers.

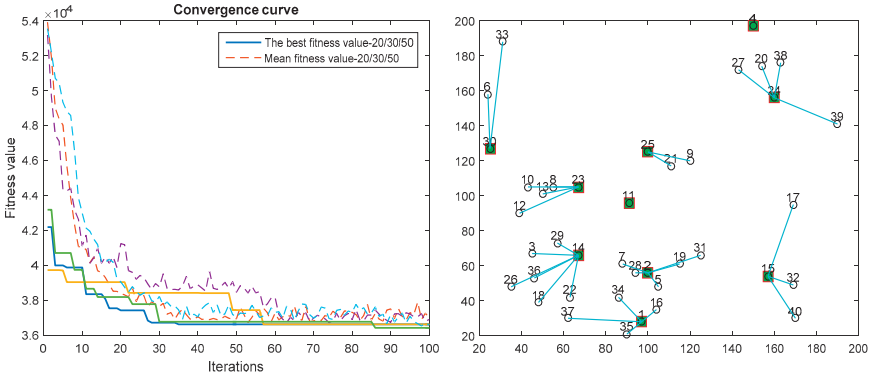


Figure 9. Convergence curves and optimal distribution centers scheme for the IGA algorithm for 10 distribution centers.

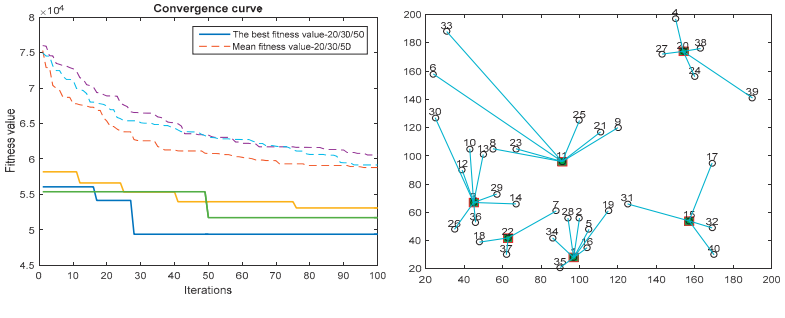


Figure 10. Convergence curves and optimal distribution centers scheme for the CS algorithm for six distribution centers.

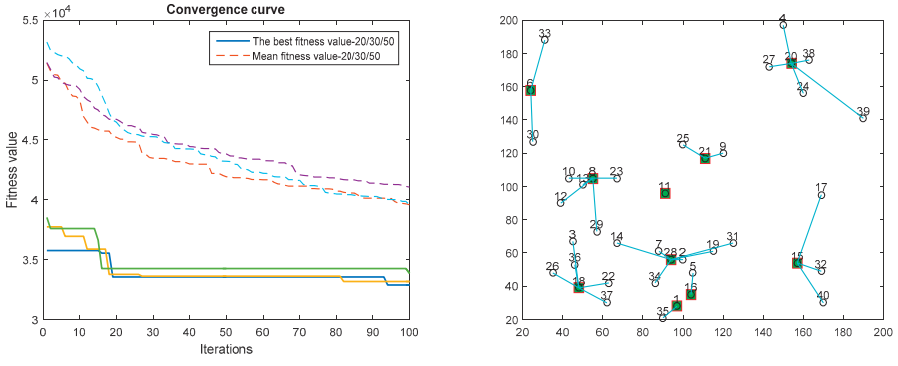


Figure 11. Convergence curves and optimal distribution centers scheme for the CS algorithm for 10 distribution centers.

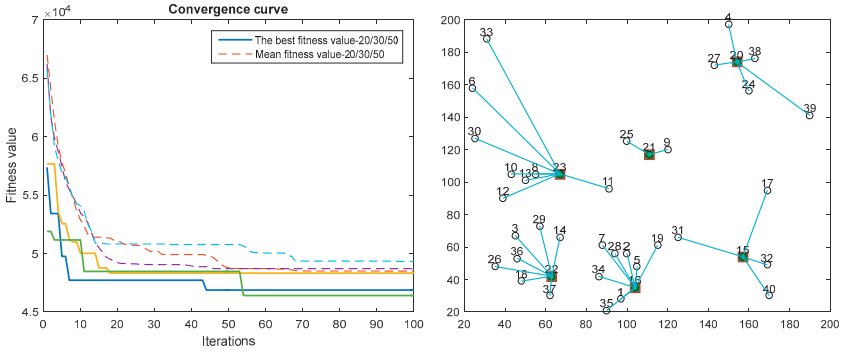


Figure 12. Convergence curves and optimal distribution centers scheme for the CCS algorithm for six distribution centers.

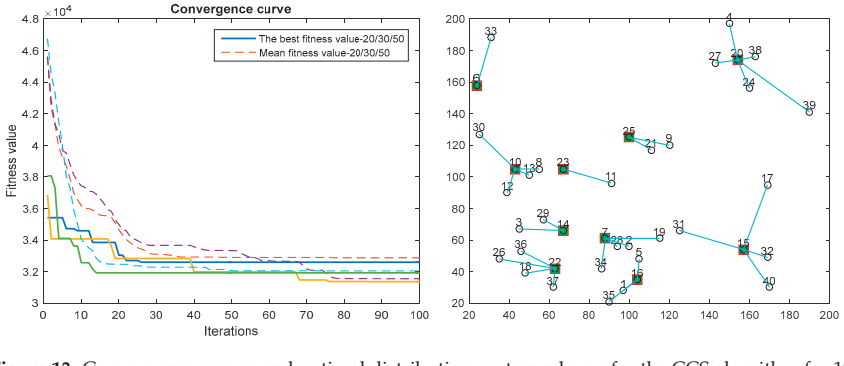


Figure 13. Convergence curves and optimal distribution centers scheme for the CCS algorithm for 10 distribution centers.

Table 17. Comparisons between DMQL-CS and other algorithms for six distribution centers.

CS		IGA		CCS	
D-C	**Distribution Scope**	**D-C**	**Distribution Scope**	**D-C**	**Distribution Scope**
3	30, 12, 10, 13, 29, 14, 36, 26	10	33, 6, 30, 12, 8, 23, 13	23	33, 6, 30, 10, 12, 13, 8, 11
11	8, 23, 6, 33, 25, 21, 9	22	26, 36, 3, 18, 29, 14, 37, 35	21	9, 25
22	18, 37, 7	21	25, 11, 9	22	26, 36, 3, 29, 14, 18, 37
1	34, 35, 28, 2, 5, 16, 19	2	7, 34, 28, 19, 31, 1, 16, 5, 19	16	1, 35, 34, 7, 28, 2, 5, 19
15	31, 32, 17, 40	20	4, 27, 24, 38, 39	15	31, 17, 32, 40
20	27, 4, 38, 24, 39	17	15, 32, 40	20	4, 27, 24, 38, 39

Table 18. Comparisons between DMQL-CS and other algorithms for 10 distribution centers.

CS		IGA		CCS	
D-C	**Distribution Scope**	**D-C**	**Distribution Scope**	**D-C**	**Distribution Scope**
6	30, 33	30	6, 33	6	33
8	10, 12, 13, 23, 19	23	12, 10, 13, 8	10	30, 12, 13, 8
18	3, 26, 36, 22, 37	14	29, 3, 26, 36, 18, 22	23	11
11	-	1	34, 37, 35, 16	14	3, 29
21	25, 9	2	7, 28,5,19,31	22	36, 26, 18, 37
28	14, 7, 34, 2, 19, 31	11	-	25	21, 9
16	5	25	21, 9	7	34, 28, 2,19
1	35	24	17, 20, 38, 39	16	1, 35, 5
20	4, 27, 38, 24, 39	15	17, 32, 40	15	31, 17, 32, 40
15	17, 32, 40	4	-	20	4, 27, 38, 24, 39

For the third set of experiments, the CS algorithm was run 20, 30 and 50 times independently for the 40 city and six distribution center example. As shown in Figure 10, both the average convergence curve and the optimal convergence curve can converge at 80 iterations. The optimal distribution cost and average distribution cost obtained by the CS algorithm are 4.9629×10^4 and 6.1392×10^4, respectively. As shown in Figure 11, average convergence curve can converge at 100 iterations for 10 distribution centers. The optimal distribution cost and average distribution cost obtained by the CS algorithm are 3.2435×10^4 and 3.9502×10^4, which indicates that logistics distribution location strategy of CS algorithm is the worst in both the optimal convergence curve and the average convergence curve. Convergence curve of CCS algorithm can converge at 20 iterations in both the optimal convergence curve and the average convergence curve. CCS algorithm is much inferior to DMQL-CS algorithm in solving accuracy for 6 and 10 distribution centers. Although the IGA algorithm can converge, it has a lot of noise for the average convergence curve. The convergence effect of IGA is worse compared with CCS algorithm. The results of standard deviation indicate that DMQL-CS has a better robustness than the other algorithms. The optimal distribution centers and distribution addressing schemes are shown in Tables 17 and 18. According to Tables 17 and 18, the optimal distribution center points found by CS algorithm for 6 and 10 distribution centers are (3, 11, 22, 1, 15, 20) and (6, 8, 18, 11, 21, 28, 16, 1, 20, 15). The optimal distribution center points found by IGA algorithm for 6 and 10 distribution centers are (10, 22, 21, 2, 20, 17) and (30, 23, 14, 1, 2, 11, 25, 24, 15, 4). The optimal distribution center points found by CCS algorithm for 6 and 10 distribution centers are (23, 22, 21, 16, 15, 20) and (6, 10, 23, 14, 22, 25, 7, 16, 15, 20).

To further analyze the effectiveness of DMQL-CS algorithm, DMQL-CS was compared with four algorithms: CS [69], CCS [68], ICS [101], and IGA [64]. The comparison results with average fitness value (Mean), the best fitness value (Best), the worst fitness value (Worst), standard deviation (Std), and running time (Time) are shown in Table 19. It can be seen that the average distribution cost of DMQL-CS at six distribution centers is 4.8060×10^4 which is 13,332 lower than CS, and the average distribution cost in 10 distribution centers is 3.0157×10^4, which is 9345 lower than CS. Therefore, DMQL-CS is obviously superior to CS algorithm. Although ICS can provide far better final results for most of the cases, it takes more execution time because of the use of more expensive exploration operation during the initial phases. For 6 and 10 distribution centers, there is not much difference

between ICS and DMQL-CS for average distribution cost, but the optimal distribution cost of DMQL-CS is significantly higher than that of ICS algorithm. Meanwhile, from the standard deviation and running time data, it can be known that DMQL-CS has better robustness. The IGA algorithm achieved the worst performance compared with other comparison algorithms, except for the CS algorithm. For the six distribution centers, the average value obtained by IGA algorithm is 5.3008×10^4, which is 4948 more than DMQL-CS. The average value obtained by IGA algorithm is 3.6460×10^4, which is 6330 more than DMQL-CS for 10 distribution centers. CCS algorithm obtains the third best performance for 6 and 10 distribution center, respectively. In summary, the results of DMQL-CS are better than the comparison algorithms in terms of optimal value, worst value, average value, or running time. The reason may be that the *Q*-learning step size strategy improves the precision of the algorithm. The crossover and mutation operator accelerate the convergence speed of the algorithm. Overall, the selection method of logistics distribution center based on cuckoo search algorithm with *Q*-Learning and genetic operation has better optimal value compared with the five other algorithms for both 6 and 10 distribution centers, which indicates that the selection strategy based on DMQL-CS has higher solution accuracy and wider range of optimization. Meanwhile, it can be seen in Table 19 that the running time of DMQL-CS is significantly lower than the four other algorithms, and the number of iterations is significantly reduced. In general, DMQL-CS algorithm can select the address of logistics distribution center more quickly and accurately compared with the comparison algorithm. Finally, we can say that our proposed algorithm interestingly outperforms other competitive algorithms in terms of convergence rate and robustness.

Table 19. Comparisons between DMQL-CS and other algorithms for 6 and 10 distribution centers in 40 cities.

Algorithm	Distribution Points	Algorithms				
		Best	**Mean**	**Worst**	**Std**	**Time (s)**
CS	6	4.9629×10^4	6.1392×10^4	7.9211×10^4	2.4874×10^5	4.5187
	10	3.2435×10^4	3.9502×10^4	4.3961×10^4	3.9872×10^5	4.5530
CCS	6	4.7913×10^4	4.9009×10^4	5.2085×10^4	4.9009×10^5	4.9486
	10	3.1619×10^4	3.3815×10^4	3.4209×10^4	3.3815×10^4	4.8706
IGA	6	5.2032×10^4	5.3008×10^4	5.3814×10^4	5.9226×10^5	4.4255
	10	3.5424×10^4	3.6460×10^4	3.6980×10^4	9.0172×10^5	4.5235
ICS	6	4.5748×10^4	4.6187×10^4	4.6919×10^4	5.8622×10^4	4.7245
	10	3.1034×10^4	3.2197×10^4	3.3113×10^4	8.0172×10^4	4.7811
DMQL-CS	6	4.5013×10^4	4.8060×10^4	4.9253×10^4	1.2763×10^4	4.6255
	10	2.9811×10^4	3.0157×10^4	3.2132×10^4	2.7651×10^4	4.6509

6. Conclusions

In this study, we constructed a model of CS with *Q*-Learning and genetic operators, and then solved the address of logistics distribution center with DMQL-CS algorithm in which adopts *Q*-Learning scheme to learn the individual optimal step size strategy according to the effect of individual multi-steps. The most appropriate step size control strategy is chosen as a parameter for the current step size evolution of the cuckoo, which increases the adaptability of individual evolution. At the same time, to accelerate the convergence of the algorithm, genetic operators and hybrid operations are added to DMQL-CS algorithm. Crossover and mutation operations expand the search area of the population, and accelerate the convergence of the DMQL-CS algorithm.

To verify the performance of DMQL-CS, DMQL-CS was employed to solve fifteen benchmark test functions and CEC 2013 test suit. The results show that the proposed DMQL-CS algorithm clearly outperforms the standard CS algorithm. Comparing with some improved CS variants and DE variants, we found that DMQL-CS algorithm outperforms the other algorithms on a majority of benchmarks.

In addition, the effectiveness of the proposed method was further verified by comparing with CS, ICS, CCS, and IGA for both 6 and 10 distribution centers.

In the future, we will focus our research work on the study of special cases to strengthen the algorithm in more complex conditions. We will determine how to generalize our work to handle combinatorial optimization problems and to extend DMQL-CS optimization algorithms to in the realistic engineering areas and feature selection for machine learning [102].

Author Contributions: Conceptualization, J.L.; methodology, H.L.; software, D.-d.X. and T.Z.; validation, T.T.; writing—original draft preparation, J.L. and H.L.; writing—review and editing, D.-d.X. and T.Z.; funding acquisition, J.L. and T.T.; All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by doctoral Foundation of Wuhan Technology and Business University (No. D2019010) and the National Natural Science Foundation of China (No. 61503220).

Acknowledgments: The authors would like to thank the anonymous reviewers and the editor for their careful reviews and constructive suggestions to help us improve the quality of this paper.

Conflicts of Interest: The authors declare that they have no conflicts of interest.

References

1. Sang, H.-Y.; Pan, Q.-K.; Duan, P.-Y.; Li, J.-Q. An effective discrete invasive weed optimization algorithm for lot-streaming flowshop scheduling problems. *J. Intell. Manuf.* **2015**, *29*, 1337–1349. [CrossRef]
2. Sang, H.-Y.; Pan, Q.-K.; Li, J.-Q.; Wang, P.; Han, Y.-Y.; Gao, K.-Z.; Duan, P. Effective invasive weed optimization algorithms for distributed assembly permutation flowshop problem with total flowtime criterion. *Swarm Evol. Comput.* **2019**, *44*, 64–73. [CrossRef]
3. Li, M.; Xiao, D.; Zhang, Y.; Nan, H. Reversible data hiding in encrypted images using cross division and additive homomorphism. *Signal Process. Image Commun.* **2015**, *39*, 234–248. [CrossRef]
4. Li, M.; Guo, Y.; Huang, J.; Li, Y. Cryptanalysis of a chaotic image encryption scheme based on permutation-diffusion structure. *Signal Process. Image Commun.* **2018**, *62*, 164–172. [CrossRef]
5. Fan, H.; Li, M.; Liu, D.; Zhang, E. Cryptanalysis of a colour image encryption using chaotic APFM nonlinear adaptive filter. *Signal Process.* **2018**, *143*, 28–41. [CrossRef]
6. Dong, W.; Shi, G.; Li, X.; Ma, Y.; Huang, F. Compressive sensing via nonlocal low-rank regularization. *IEEE Trans. Image Process.* **2014**, *23*, 3618–3632. [CrossRef]
7. Zhang, Y.; Gong, D.; Hu, Y.; Zhang, W. Feature selection algorithm based on bare bones particle swarm optimization. *Neurocomputin* **2015**, *148*, 150–157. [CrossRef]
8. Zhang, Y.; Song, X.-F.; Gong, D.-W. A return-cost-based binary firefly algorithm for feature selection. *Inf. Sci.* **2017**, *418*, 561–574. [CrossRef]
9. Mao, W.; He, J.; Tang, J.; Li, Y. Predicting remaining useful life of rolling bearings based on deep feature representation and long short-term memory neural network. *Adv. Mech. Eng.* **2018**, *10*, 1687814018817184. [CrossRef]
10. Jian, M.; Lam, K.-M.; Dong, J. Facial-feature detection and localization based on a hierarchical scheme. *Inf. Sci.* **2014**, *262*, 1–14. [CrossRef]
11. Wang, G.-G.; Chu, H.E.; Mirjalili, S. Three-dimensional path planning for UCAV using an improved bat algorithm. *Aerosp. Sci. Technol.* **2016**, *49*, 231–238. [CrossRef]
12. Wang, G.; Guo, L.; Duan, H.; Liu, L.; Wang, H.; Shao, M. Path planning for uninhabited combat aerial vehicle using hybrid meta-heuristic DE/BBO algorithm. *Adv. Sci. Eng. Med.* **2012**, *4*, 550–564. [CrossRef]
13. Zhang, Y.; Gong, D.-w.; Gao, X.-z.; Tian, T.; Sun, X.-y. Binary differential evolution with self-learning for multi-objective feature selection. *Inf. Sci.* **2020**, *507*, 67–85. [CrossRef]
14. Wang, G.-G.; Cai, X.; Cui, Z.; Min, G.; Chen, J. High performance computing for cyber physical social systems by using evolutionary multi-objective optimization algorithm. *IEEE Trans. Emerg. Top. Comput.* **2017**. [CrossRef]
15. Cui, Z.; Sun, B.; Wang, G.-G.; Xue, Y.; Chen, J. A novel oriented cuckoo search algorithm to improve DV-Hop performance for cyber-physical systems. *J. Parallel Distrib. Comput.* **2017**, *103*, 42–52. [CrossRef]
16. Jian, M.; Lam, K.-M.; Dong, J. Illumination-insensitive texture discrimination based on illumination compensation and enhancement. *Inf. Sci.* **2014**, *269*, 60–72. [CrossRef]

17. Jian, M.; Lam, K.M.; Dong, J.; Shen, L. Visual-patch-attention-aware saliency detection. *IEEE Trans. Cybern.* **2015**, *45*, 1575–1586. [CrossRef]
18. Wang, G.-G.; Lu, M.; Dong, Y.-Q.; Zhao, X.-J. Self-adaptive extreme learning machine. *Neural Comput. Appl.* **2016**, *27*, 291–303. [CrossRef]
19. Mao, W.; Zheng, Y.; Mu, X.; Zhao, J. Uncertainty evaluation and model selection of extreme learning machine based on Riemannian metric. *Neural Comput. Appl.* **2013**, *24*, 1613–1625. [CrossRef]
20. Liu, G.; Zou, J. Level set evolution with sparsity constraint for object extraction. *IET Image Process.* **2018**, *12*, 1413–1422. [CrossRef]
21. Rizk-Allah, R.M.; El-Sehiemy, R.A.; Deb, S.; Wang, G.-G. A novel fruit fly framework for multi-objective shape design of tubular linear synchronous motor. *J. Supercomput.* **2017**, *73*, 1235–1256. [CrossRef]
22. Yi, J.-H.; Xing, L.-N.; Wang, G.-G.; Dong, J.; Vasilakos, A.V.; Alavi, A.H.; Wang, L. Behavior of crossover operators in NSGA-III for large-scale optimization problems. *Inf. Sci.* **2020**, *509*, 470–487. [CrossRef]
23. Yi, J.-H.; Deb, S.; Dong, J.; Alavi, A.H.; Wang, G.-G. An improved NSGA-III Algorithm with adaptive mutation operator for big data optimization problems. *Future Gener. Comput. Syst.* **2018**, *88*, 571–585. [CrossRef]
24. Sun, J.; Miao, Z.; Gong, D.; Zeng, X.-J.; Li, J.; Wang, G.-G. Interval multi-objective optimization with memetic algorithms. *IEEE Trans. Cybern.* **2019**. [CrossRef] [PubMed]
25. Feng, Y.; Wang, G.-G. Binary moth search algorithm for discounted {0-1} knapsack problem. *IEEE Access* **2018**, *6*, 10708–10719. [CrossRef]
26. Feng, Y.; Wang, G.-G.; Wang, L. Solving randomized time-varying knapsack problems by a novel global firefly algorithm. *Eng. Comput.* **2018**, *34*, 621–635. [CrossRef]
27. Abdel-Basset, M.; Zhou, Y. An elite opposition-flower pollination algorithm for a 0-1 knapsack problem. *Int. J. Bio-Inspired Comput.* **2018**, *11*, 46–53. [CrossRef]
28. Yi, J.-H.; Wang, J.; Wang, G.-G. Improved probabilistic neural networks with self-adaptive strategies for transformer fault diagnosis problem. *Adv. Mech. Eng.* **2016**, *8*, 1–13. [CrossRef]
29. Mao, W.; He, J.; Li, Y.; Yan, Y. Bearing fault diagnosis with auto-encoder extreme learning machine: A comparative study. *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.* **2016**, *231*, 1560–1578. [CrossRef]
30. Mao, W.; Feng, W.; Liang, X. A novel deep output kernel learning method for bearing fault structural diagnosis. *Mech. Syst. Signal Process.* **2019**, *117*, 293–318. [CrossRef]
31. Duan, H.; Zhao, W.; Wang, G.; Feng, X. Test-sheet composition using analytic hierarchy process and hybrid metaheuristic algorithm TS/BBO. *Math. Probl. Eng.* **2012**, *2012*, 1–22. [CrossRef]
32. Wang, G.G.; Tan, Y. Improving metaheuristic algorithms with information feedback models. *IEEE Trans. Cybern.* **2019**, *49*, 542–555. [CrossRef] [PubMed]
33. Rong, M.; Gong, D.; Zhang, Y.; Jin, Y.; Pedrycz, W. Multidirectional prediction approach for dynamic multiobjective optimization problems. *IEEE Trans. Cybern.* **2019**, *49*, 3362–3374. [CrossRef] [PubMed]
34. Gong, D.; Sun, J.; Miao, Z. A Set-Based Genetic Algorithm for Interval Many-Objective Optimization Problems. *IEEE Trans. Evol. Comput.* **2018**, *22*, 47–60. [CrossRef]
35. Gong, D.; Sun, J.; Ji, X. Evolutionary algorithms with preference polyhedron for interval multi-objective optimization problems. *Inf. Sci.* **2013**, *233*, 141–161. [CrossRef]
36. Sergeyev, Y.D.; Kvasov, D.E.; Mukhametzhanov, M.S. On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget. *Sci. Rep.* **2018**, *8*, 453. [CrossRef]
37. Kvasov, D.E.; Mukhametzhanov, M.S. Metaheuristic vs. deterministic global optimization algorithms: The univariate case. *Appl. Math. Comput.* **2018**, *318*, 245–259. [CrossRef]
38. Sergeyev, Y.D.; Kvasov, D.E.; Mukhametzhanov, M.S. Operational zones for comparing metaheuristic and deterministic one-dimensional global optimization algorithms. *Math. Comput. Simul.* **2017**, *141*, 96–109. [CrossRef]
39. Yang, X.S.; Gandomi, A.H. Bat algorithm: A novel approach for global engineering optimization. *Eng. Comput.* **2012**, *29*, 464–483. [CrossRef]
40. Simon, D. Biogeography-based optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [CrossRef]
41. Dorigo, M.; Stutzle, T. *Ant Colony Optimization*; MIT Press: Cambridge, MA, USA, 2004.
42. Wang, G.-G.; Deb, S.; Coelho, L.d.S. Earthworm optimization algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Int. J. Bio-Inspired Comput.* **2018**, *12*, 1–22. [CrossRef]

43. Wang, G.-G.; Deb, S.; Coelho, L.d.S. Elephant herding optimization. In Proceedings of the 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI), Bali, Indonesia, 7–9 December 2015; pp. 1–5.
44. Wang, G.-G.; Deb, S.; Gao, X.-Z.; Coelho, L.d.S. A new metaheuristic optimization algorithm motivated by elephant herding behavior. *Int. J. Bio-Inspired Comput.* **2016**, *8*, 394–409. [CrossRef]
45. Wang, G.-G. Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Memetic Comput.* **2018**, *10*, 151–164. [CrossRef]
46. Yang, X.S. Firefly algorithm, stochastic test functions and design optimisation. *Int. J. Bio-Inspired Comput.* **2010**, *2*, 78–84. [CrossRef]
47. Wang, H.; Yi, J.-H. An improved optimization method based on krill herd and artificial bee colony with information exchange. *Memetic Comput.* **2018**, *10*, 177–198. [CrossRef]
48. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
49. Liu, F.; Sun, Y.; Wang, G.-G.; Wu, T. An artificial bee colony algorithm based on dynamic penalty and chaos search for constrained optimization problems. *Arab. J. Sci. Eng.* **2018**, *43*, 7189–7208. [CrossRef]
50. Geem, Z.W.; Kim, J.H.; Loganathan, G.V. A new heuristic optimization algorithm: Harmony search. *Simulation* **2001**, *76*, 60–68. [CrossRef]
51. Wang, G.-G.; Gandomi, A.H.; Zhao, X.; Chu, H.E. Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. *Soft Comput.* **2016**, *20*, 273–285. [CrossRef]
52. Wang, G.-G.; Deb, S.; Cui, Z. Monarch butterfly optimization. *Neural Comput. Appl.* **2019**, *31*, 1995–2014. [CrossRef]
53. Wang, G.-G.; Deb, S.; Zhao, X.; Cui, Z. A new monarch butterfly optimization with an improved crossover operator. *Oper. Res. Int. J.* **2018**, *18*, 731–755. [CrossRef]
54. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
55. Sun, Y.; Jiao, L.; Deng, X.; Wang, R. Dynamic network structured immune particle swarm optimisation with small-world topology. *Int. J. Bio-Inspired Comput.* **2017**, *9*, 93–105. [CrossRef]
56. Gandomi, A.H.; Alavi, A.H. Multi-stage genetic programming: A new strategy to nonlinear system modeling. *Inf. Sci.* **2011**, *181*, 5227–5239. [CrossRef]
57. Gandomi, A.H.; Alavi, A.H. Krill herd: A new bio-inspired optimization algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **2012**, *17*, 4831–4845. [CrossRef]
58. Wang, G.-G.; Gandomi, A.H.; Alavi, A.H. An effective krill herd algorithm with migration operator in biogeography-based optimization. *Appl. Math. Model.* **2014**, *38*, 2454–2462. [CrossRef]
59. Wang, G.-G.; Gandomi, A.H.; Alavi, A.H.; Deb, S. A multi-stage krill herd algorithm for global numerical optimization. *Int. J. Artif. Intell. Tool* **2016**, *25*, 1550030. [CrossRef]
60. Wang, G.-G.; Gandomi, A.H.; Alavi, A.H.; Gong, D. A comprehensive review of krill herd algorithm: Variants, hybrids and applications. *Artif. Intell. Rev.* **2019**, *51*, 119–148. [CrossRef]
61. Wang, G.-G.; Guo, L.; Gandomi, A.H.; Hao, G.-S.; Wang, H. Chaotic krill herd algorithm. *Inf. Sci.* **2014**, *274*, 17–34. [CrossRef]
62. Wang, G.-G.; Deb, S.; Gandomi, A.H.; Alavi, A.H. Opposition-based krill herd algorithm with Cauchy mutation and position clamping. *Neurocomputing* **2016**, *177*, 147–157. [CrossRef]
63. Wang, G.-G.; Gandomi, A.H.; Alavi, A.H. Stud krill herd algorithm. *Neurocomputing* **2014**, *128*, 363–370. [CrossRef]
64. Wang, X.; Zhang, X.J.; Cao, X.; Zhang, J.; Fen, L. An improved genetic algorithm based on immune principle. *Minimicro Syst.* **1999**, *20*, 120.
65. Li, J.; Li, Y.-x.; Tian, S.-s.; Zou, J. Dynamic cuckoo search algorithm based on Taguchi opposition-based search. *Int. J. Bio-Inspired Comput.* **2019**, *13*, 59–69. [CrossRef]
66. Yang, X.S.; Deb, S. Engineering optimisation by cuckoo search. *Int. J. Math. Model. Numer. Optim.* **2010**, *1*, 330–343. [CrossRef]
67. Gandomi, A.H.; Yang, X.-S.; Alavi, A.H. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Eng. Comput.* **2013**, *29*, 17–35. [CrossRef]
68. Wang, G.-G.; Deb, S.; Gandomi, A.H.; Zhang, Z.; Alavi, A.H. Chaotic cuckoo search. *Soft Comput.* **2016**, *20*, 3349–3362. [CrossRef]

69. Yang, X.-S.; Deb, S. Cuckoo search via Lévy flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214.
70. Mlakar, U.; Fister, I.; Fister, I. Hybrid self-adaptive cuckoo search for global optimization. *Swarm Evol. Comput.* **2016**, *29*, 47–72. [CrossRef]
71. Li, X.; Wang, J.; Yin, M. Enhancing the performance of cuckoo search algorithm using orthogonal learning method. *Neural Comput. Appl.* **2013**, *24*, 1233–1247. [CrossRef]
72. Ouaarab, A.; Ahiod, B.; Yang, X.-S. Discrete cuckoo search algorithm for the travelling salesman problem. *Neural Comput. Appl.* **2013**, *24*, 1659–1669. [CrossRef]
73. Wang, Y.-H.; Li, T.-H.S.; Lin, C.-J. Backward Q-learning: The combination of Sarsa algorithm and Q-learning. *Eng. Appl. Artif. Intell.* **2013**, *26*, 2184–2193. [CrossRef]
74. Alexandridis, A.; Chondrodima, E.; Sarimveis, H. Cooperative learning for radial basis function networks using particle swarm optimization. *Appl. Soft Comput.* **2016**, *49*, 485–497. [CrossRef]
75. Rakhshani, H.; Rahati, A. Snap-drift cuckoo search: A novel cuckoo search optimization algorithm. *Appl. Soft Comput.* **2017**, *52*, 771–794. [CrossRef]
76. Samma, H.; Lim, C.P.; Saleh, J.M. A new reinforcement learning-based memetic particle swarm optimizer. *Appl. Soft Comput.* **2016**, *43*, 276–297. [CrossRef]
77. Abed-alguni, B.H. Action-selection method for reinforcement learning based on cuckoo search algorithm. *Arab. J. Sci. Eng.* **2017**, *43*, 6771–6785. [CrossRef]
78. Ma, H.-S.; Li, S.-X.; Li, S.-F.; Lv, Z.-N.; Wang, J.-S. An improved dynamic self-adaption cuckoo search algorithm based on collaboration between subpopulations. *Neural Comput. Appl.* **2018**, *31*, 1375–1389. [CrossRef]
79. Li, X.; Yin, M. Modified cuckoo search algorithm with self adaptive parameter method. *Inf. Sci.* **2015**, *298*, 80–97. [CrossRef]
80. Yang, B.; Miao, J.; Fan, Z.; Long, J.; Liu, X. Modified cuckoo search algorithm for the optimal placement of actuators problem. *Appl. Soft Comput.* **2018**, *67*, 48–60. [CrossRef]
81. Li, J.; Li, Y.-X.; Tian, S.-S.; Xia, J.-L. An improved cuckoo search algorithm with self-adaptive knowledge learning. *Neural Comput. Appl.* **2019**. [CrossRef]
82. Cheng, J.; Wang, L.; Xiong, Y. Ensemble of cuckoo search variants. *Comput. Ind. Eng.* **2019**, *135*, 299–313. [CrossRef]
83. Long, W.; Cai, S.; Jiao, J.; Xu, M.; Wu, T. A new hybrid algorithm based on grey wolf optimizer and cuckoo search for parameter extraction of solar photovoltaic models. *Energy Convers. Manag.* **2020**, *203*, 112243. [CrossRef]
84. Zhang, Z.; Ding, S.; Jia, W. A hybrid optimization algorithm based on cuckoo search and differential evolution for solving constrained engineering problems. *Eng. Appl. Artif. Intell.* **2019**, *85*, 254–268. [CrossRef]
85. Zhang, X.; Li, X.-T.; Yin, M.-H. Hybrid cuckoo search algorithm with covariance matrix adaption evolution strategy for global optimisation problem. *Int. J. Bio-Inspired Comput.* **2019**, *13*, 102–110. [CrossRef]
86. Tang, H.; Xue, F. Cuckoo search algorithm with different distribution strategy. *Int. J. Bio-Inspired Comput.* **2019**, *13*, 234–241. [CrossRef]
87. Ong, P.; Zainuddin, Z. Optimizing wavelet neural networks using modified cuckoo search for multi-step ahead chaotic time series prediction. *Appl. Soft Comput.* **2019**, *80*, 374–386. [CrossRef]
88. Kamoona, A.M.; Patra, J.C. A novel enhanced cuckoo search algorithm for contrast enhancement of gray scale images. *Appl. Soft Comput.* **2019**, *85*, 105749. [CrossRef]
89. Naidu, M.N.; Boindala, P.S.; Vasan, A.; Varma, M.R. Optimization of Water Distribution Networks Using Cuckoo Search Algorithm. In *Advanced Engineering Optimization Through Intelligent Techniques*; Springer: Singapore, 2020.
90. Ibrahim, A.M.; Tawhid, M.A. A hybridization of cuckoo search and particle swarm optimization for solving nonlinear systems. *Evol. Intell.* **2019**, *12*, 541–561. [CrossRef]
91. Osaba, E.; del Ser, J.; Camacho, D.; Bilbao, M.N.; Yang, X.-S. Community detection in networks using bio-inspired optimization: Latest developments, new results and perspectives with a selection of recent meta-heuristics. *Appl. Soft Comput.* **2020**, *87*, 106010. [CrossRef]
92. Rath, A.; Samantaray, S.; Swain, P.C. Optimization of the Cropping Pattern Using Cuckoo Search Technique. In *Smart Techniques for a Smarter Planet*; Springer: Cham, Switzerland, 2019; p. 19.

93. Abdel-Basset, M.; Wang, G.-G.; Sangaiah, A.K.; Rushdy, E. Krill herd algorithm based on cuckoo search for solving engineering optimization problems. *Multimed. Tools Appl.* **2017**, *78*, 3861–3884. [CrossRef]
94. Cao, Z.; Lin, C.; Zhou, M.; Huang, R. Scheduling semiconductor testing facility by using cuckoo search algorithm with reinforcement learning and surrogate modeling. *IEEE Trans. Autom. Sci. Eng.* **2019**, *16*, 825–837. [CrossRef]
95. Hu, H.; Li, X.; Zhang, Y.; Shang, C.; Zhang, S. Multi-objective location-routing model for hazardous material logistics with traffic restriction constraint in inter-city roads. *Comput. Ind. Eng.* **2019**, *128*, 861–876. [CrossRef]
96. Wang, F.; He, X.-s.; Wang, Y. The cuckoo search algorithm based on Gaussian disturbance. *J. Xi'an Polytech. Univ.* **2011**, *25*, 566–569.
97. Wang, F.; Luo, L.; He, X.S.; Wang, Y. Hybrid optimization algorithm of PSO and Cuckoo Search. In Proceedings of the 2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), Dengleng, China, 8–10 August 2011; IEEE: Piscataway, NJ, USA, 2011.
98. Brest, J.; Greiner, S.; Boskovic, B.; Mernik, M.; Zumer, V. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Trans. Evol. Comput.* **2006**, *10*, 646–657. [CrossRef]
99. Qin, A.K.; Huang, V.L.; Suganthan, P.N. Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization. *IEEE Trans. Evol. Comput.* **2009**, *13*, 398–417. [CrossRef]
100. Liang, J.J.; Qin, A.K.; Suganthan, P.N.; Baskar, S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evol. Comput.* **2006**, *10*, 281–295. [CrossRef]
101. Zhao, S.A.; Qu, C.W. An Improved Cuckoo Algorithm for Solving the Problem of Logistics Distribution Center Location. *Math. Pract. Theory* **2017**, *47*, 206–213.
102. del Ser, J.; Osaba, E.; Molina, D.; Yang, X.-S.; Salcedo-Sanz, S.; Camacho, D.; Das, S.; Suganthan, P.N.; Coello, C.A.C.; Herrera, F. Bio-inspired computation: Where we stand and what's next. *Swarm Evol. Comput.* **2019**, *48*, 220–250. [CrossRef]

mathematics

MDPI

Article

A Comparison of Evolutionary and Tree-Based Approaches for Game Feature Validation in Real-Time Strategy Games with a Novel Metric

Damijan Novak *, Domen Verber, Jani Dugonik and Iztok Fister, Jr.

Institute of Informatics, University of Maribor, Koroška Cesta 46, 2000 Maribor, Slovenia; domen.verber@um.si (D.V.); jani.dugonik@um.si (J.D.); iztok.fister1@um.si (I.F.J.)
* Correspondence: damijan.novak@um.si; Tel.: +386-2-220-7431

Received: 2 April 2020; Accepted: 29 April 2020; Published: 1 May 2020

Abstract: When it comes to game playing, evolutionary and tree-based approaches are the most popular approximate methods for decision making in the artificial intelligence field of game research. The evolutionary domain therefore draws its inspiration for the design of approximate methods from nature, while the tree-based domain builds an approximate representation of the world in a tree-like structure, and then a search is conducted to find the optimal path inside that tree. In this paper, we propose a novel metric for game feature validation in Real-Time Strategy (RTS) games. Firstly, the identification and grouping of Real-Time Strategy game features is carried out, and, secondly, groups are included into weighted classes with regard to their correlation and importance. A novel metric is based on the groups, weighted classes, and how many times the playtesting agent invalidated the game feature in a given game feature scenario. The metric is used in a series of experiments involving recent state-of-the-art evolutionary and tree-based playtesting agents. The experiments revealed that there was no major difference between evolutionary-based and tree-based playtesting agents.

Keywords: evolutionary computation; playtesting; game feature; game simulation; game trees; playtesting metric; validation

1. Introduction

Real-Time Strategy (RTS) games are designed as turn-based games where players, each following their own strategies, try to defeat one another through a series of turns. The term 'strategy' stands for the highest form of decision-making process, where the ultimate purpose is to defeat the opponent. Decisions are made between turns (a turn is a transition from the current game state to the next one), which are so short (i.e., in the range of milliseconds) that the game looks as though it is progressing in real time. After a decision is made, the actions are executed. The difference between RTS games and classical turn-based board games, of which probably the most well-known representative is the game of chess, is in the execution of the actions. Actions in RTS games are durative and simultaneous [1], as opposed to the instant moves, of which each player can make one per turn, in classical board games.

During the last decade, RTS games have become one of the best test beds for researching Artificial Intelligence (AI) for games [2,3]. The main reason for the growth in research is the fact that RTS games offer plenty of challenges for researchers. For example, RTS games are representatives of the highest class of computational complexity [4], which is due to their extremely large state-action spaces [5] (i.e., search space). Search space is often impossible to search exhaustively, because a specific game is a high-dimensional space of game variants (many different parameters are available), and it is also called game space [6].

Exploring the search space of games is often considered to be a difficult problem [7], and most of the complex optimization problems relating to games' search spaces cannot be solved using the exact methods that search for the optimal solution by enumerating all possible solutions. To solve these problems, various methods have emerged in the past decades that solve problems approximately. In recent times, researchers have been looking for inspiration for the design of these approximate algorithms/methods in nature, e.g., Darwin's evolutionary theory [8], the collective behavior of social living insects [9], the social behavior of some animal species [10,11], physical phenomena [12], and so on.

The bio-inspired computation field [13] is a field that covers all of the algorithms/methods that fall within the scope of these mentioned inspirations and is an extensively studied research area of AI. Nowadays, numerous algorithms exist that fall under the bio-inspired computation umbrella, such as the Artificial Bee Colony (ABC) Algorithm [14], Differential Evolution (DE) [15], Firefly Algorithm (FA), Genetic Algorithm (GA) [16], Monarch Butterfly Optimization (MBO) [17], etc. Due to the popularity of this subject, numerous unprecedented implications of these approaches exist among real-world applications [13]. Some of the application areas where bio-inspired computation approaches have been successfully applied include: antenna design [18], medicine [19], and dynamic data stream clustering [20].

In addition to the many different application areas, bio-inspired computation also plays an important role in the design and development of games. Bio-inspired computation approaches in games have been used for procedural content generation [21], the development of controllers that are able to play games [22], educational and serious games [23], intelligent gaming systems [24], evolutionary methods in board games [25], behavioral design of non-player characters [26], etc.

Gameplaying agents (algorithms) are made to play the game in question, with the game rules being hard-coded or self-obtained (general gameplaying), in a self-sustained way (i.e., no human input is needed during the (general) gameplay) [27]. The primary task of the gameplaying agent is to win games, and the secondary task is to win them with a higher score [28]. For the RTS gameplaying agent [29] to be able to cope with the high computational complexity of the game space, it has to be able to function inside different segments of the game, which are as follows: resource and production management (also categorized as economy) [30], strategical [31], tactical [32] and micromanagement [33] operations, scouting [34] and sometimes even diplomacy [35]. For one to be successful when playing an RTS game, a balanced combination of all those segments must be considered by the agent [36]. Since gameplaying agents are already built to operate and cover a variety of tasks in a given game space, they can be adapted to become playtesting agents.

Playtesting agents are meant to play through the game (or a slice of it) and try to explore the behavior that can generate data that would assist developers during the development phase of a game [37,38]. Game studios conduct countless tests on gameplaying with real players [39], but relying on humans for playtesting can result in higher costs and can also be inefficient [37]. The research on playtesting is, therefore, very important for the following two reasons: it has a huge economic potential and is of considerable interest to the game industry [40]. Further economic potential is also apparent in semi-related fields, like Gamification [41].

A Game Design Document (GDC) specifies core gameplay, game elements, necessary game features, etc. [42]. With this paper, we tackle the problem of the automatic validation of game features for the game space specified in GDC and also address research requirements from articles (for instance, [43]), where the authors point out the need that games with a higher complexity have of scaling.

In this article, we will try to find the answers to the following research questions:

- RQ1: How easy is it to adapt gameplaying agents as playtesting agents in RTS games?
- RQ2: Which RTS game definitions can be used to make a comparison between different playtesting agents?
- RQ3: How to evaluate playtesting agents based on RTS game definitions, and which are the most beneficial to them?

- RQ4: Is there a difference between evolutionary and the non-evolutionary approaches (like standard Monte Carlo tree searches [44]) with regard to playtesting abilities?
- RQ5: How does one define valid/invalid game features in the game space?

Altogether, the main contributions of this paper are as follows:

- A novel metric is proposed to make a comparison between different playtesting agents;
- A method is proposed for adapting gameplaying agents as playtesting agents in real-time strategy games; and
- The proposed metric is used in a series of experiments involving adapted evolutionary and tree-based state-of-the-art gameplaying agents.

The structure of the remainder of this paper is as follows. Section 2 outlines the game features of real-time strategy games and the microRTS simulation environment, while Section 3 presents the proposed novel metric that will allow for the comparison of different playtesting agents. Section 4 describes the experimental environment, adaptation of gameplaying agents as playtesting agents (including detailed descriptions of them) and the results of the experiments. A Discussion is provided in Section 5, and the conclusion is presented in Section 6.

2. Real-Time Strategy Games

This chapter briefly outlines the game features of RTS games, and a description of the microRTS environment is also provided.

2.1. Game Features of RTS Games

"Game feature" is a generic term used to refer to differences and similarities between games [45]. Game features are defined in GDC [46], and, after they are implemented, each game's features rely on the use of game mechanics. Game mechanics are methods invoked by agents in interacting with the game world (e.g., to obtain the health value of the unit) [47]. In [48], 18 general definitions of game features (hereinafter referred to as groups) can be found.

In the RTS game domain, different kinds of game feature subset groupings are possible (Economic, Military, Map Coverage, Micro Skill and Macro Skill) [49], but to the best of our knowledge, the RTS game features have not yet been placed into groups. The placement of RTS game features into groups is, in our opinion, important, because it allows for the possibility of comparing RTS game features with the features of other game genres in the future.

2.2. microRTS

There are many different RTS game worlds in existence. Not all of them are openly available, but even some of the commercial ones have been opened up for research purposes (e.g., StarCraft™ was opened through the programming interface). microRTS is a simple non-commercial simulation environment, which was created to test any theoretical ideas a game researcher might have.

This simulation environment follows standard RTS game genre game rules:

1. Players gather resources and use them to create structures and new mobile units;
2. The game goal is to defeat the opposing player in a battle for supremacy; and
3. Resources, structures, and mobile units must be cleverly used.

The microRTS environment includes the following features (seen in Figure 1):

- Four mobile units: worker, light (battle unit), heavy (battle unit) and ranged (battle unit);
- Two structures: base and barracks;
- Resources; and
- A wall.

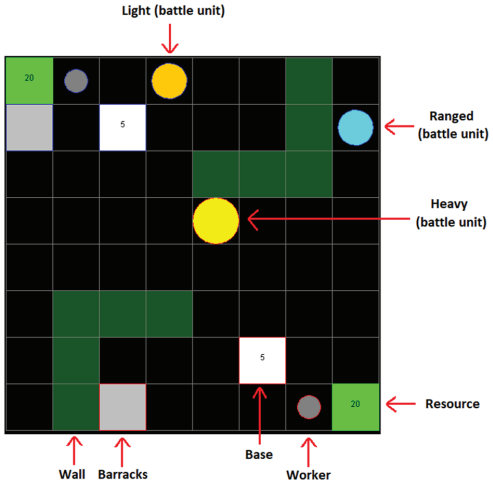


Figure 1. Micro real-time strategy (microRTS) environment, with all features visible.

Workers are used to gather resources and build structures, and they also possess the ability to attacks with limited firepower. Light, heavy and ranged are the main battle units used for attacks on opponent structures and mobile units. Battle units have different initial properties (i.e., a heavy battle unit can sustain more damage before being destroyed versus a light battle unit, and a ranged unit can shoot farther). Bases produce workers. Barracks are used to create battle units. The wall is used as a physical barrier in the map.

microRTS allows configurable scenarios to be placed in the environment. Figure 1 presents one such scenario: an 8×8 cell map, with fixed positions for resources (in the top left and bottom right corners) and walls. The mobile units are not fixed and can be moved freely inside this environment.

Scenarios can be configured for varying map sizes (4×4, 8×8, 12×12, etc.) and with different starting positions for the unit types, structures, and resources (which can be placed anywhere on the map). The game can be played with visible features (graphical interface turned on for observations) or in the background (which allows for a faster execution of scenarios and quicker overall simulations, with less computer resources used).

microRTS also already includes many gameplaying agents that can be used in experiments.

3. Proposal of a Metric for Game Feature Validation

Our motivation to create a metric came from the need to be able to differentiate easily between different playtesting agents' performances, when multiple game features need to be validated. In order to propose a novel metric for comparing playtesting agents, the following steps were considered in our study:

- STEP 1: The RTS game features are identified;
- STEP 2: The game features are grouped in precise game feature groups; (STEP 2.1): Classification of game feature groups according to their correlation (groups that are similar in description tend to be correlated, and this also allows single game features to be placed into multiple groups) and importance (some groups are of a higher importance, because they reflect and are essential to RTS gameplay, while some could be left out without jeopardizing the game's position in the RTS game genre);
- STEP 3: For empty groups in STEP 2, a further identification of the RTS game features is conducted by including more search strings and other search engines (e.g., Google Scholar); and
- STEP 4: The novel metric is proposed.

All steps are described in detail in the following subsections and are presented graphically in Figure 2.

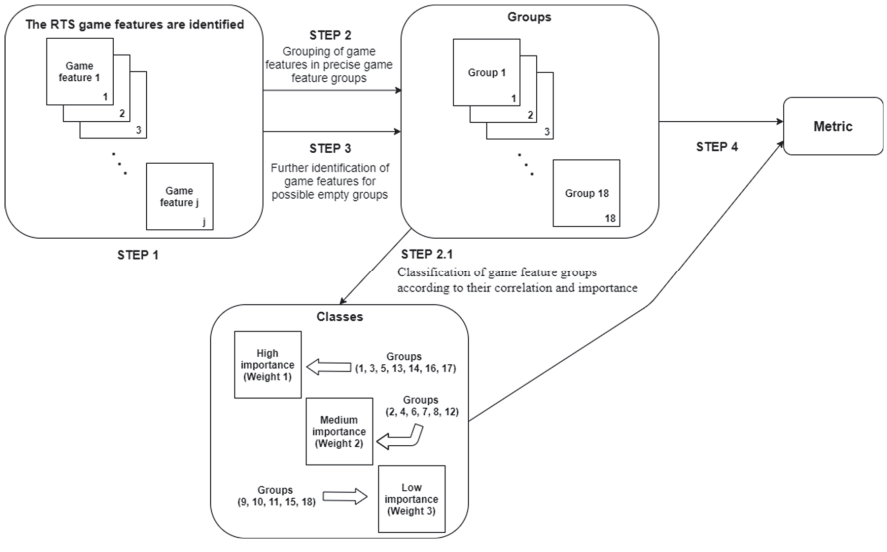


Figure 2. Pathway (steps taken) from the identification of the RTS game features to the novel metric.

3.1. Identification of RTS Game Features

Game features are mentioned in many RTS game research works, but they are scattered across different subdomains and research agendas. Our goal was to use the pool of research articles and dissertations and to identify the game features included in this research. The pool was reviewed with the help of a literature search. The ISI Web of Science and ProQuest research search engines were used. A search query with the following search string was made: "game features" and "real-time strategy games", which returned 88 hits for the ISI Web of Science and 34 hits for ProQuest.

The results (articles and dissertations) were filtered to exclude non-RTS game research works. A manual search was conducted through the research work for mentions of the "feature" string (note: 14 works from the ISI Web of Science and 0 for ProQuest were located after a manual search). The located text was extracted and analyzed for surrounding context, then transformed into a compact format that could act as a short game feature description. The surrounding context was used to transform the text, because not all research work has game features that can be used as-is. A short description was then made of the list of game feature descriptions. If a description was already on the list, and it was not adding additional information, it was omitted (note: seven works from the ISI Web of Science were omitted). Additionally, one research work can include more than just one mention of the string "feature" with the surrounding context.

Note: Future work could broaden the scope and include other search strings (like "aspect" or "feature") for a more in-depth survey of the general RTS features.

Table 1 includes a list of short game feature descriptions, which were produced after the completion of the first step. The short game feature descriptions are accompanied by a reference.

Table 1. Game feature descriptions derived from related work.

Game Feature Label	Short Game Feature Description	Short Game Feature Label	Reference (Used as a Basis for Extraction)
Resource gathering	Game unit (worker) collects at least *x* units of type *A* resources and at least *y* units of type *B* resources in num trips.	GF1_RG	[50]
Game engine features and objects	Game unit (battle unit) always hits with *x* points of damage.	GF2_EOBJ	[51]
Game difficulty (aiding)	The opponent is aided with *x* more units, resulting in a player losing every game. Note: such a feature can be part of an advanced mode, where non-advanced users must not/cannot win.	GF3_DIFA	[52]
Game objective (construction)	If the player tries to, it must be able to create *x* game structure(s) (e.g., barracks).	GF4_CONS	[53]
Game assessment	Game score is calculated based on raw features (e.g., no. of workers) and must represent the game state status correctly when presented to the player.	GF5_AST	[54]
Stumbling block	The player cannot destroy the enemy in a specific part of the map due to stumbling blocks (e.g., a wall).	GF6_SB	[55]
Game exploration (unlocking new technologies)	If the player tries discovery, it can create *x* game units (e.g., battle unit–light) through the usage of game structure(s) (e.g., barracks).	GF7_EXPL	[56]
Special unit	The player is confronted with a special game unit (e.g., Super-Heavy with special features), which cannot be destroyed with the given resources.	GF8_FANT	[57]
Partial information (fog-of-war)	The player cannot operate in a partially observable environment, so it therefore cannot destroy the opponent in such an environment.	GF9_PARI	[58]
Game difficulty (challenge)	The player cannot destroy *x* structures (e.g., barracks) guarded with *y* rushing game units (e.g., battle unit–heavy) with access to *z* units of *A* type resources.	GF10_DIFC	[52]
Game control (take over the map)	The player can destroy all the structures on the map before the time runs out.	GF11_GCMP	[59]
Interaction on a complex map	If the player controlling *x* battle units (e.g., a heavy battle unit) finds a static unit (e.g., barracks) in a maze (or complex map), the static unit is always destroyed.	GF12_INTE	[60]
Resource gathering under attack	A gatherer (e.g., a worker) is always destroyed when trying to gather resources.	GF13_RG2	[61]

3.2. Grouping the Game Features into Specific Groups

The grouping of game features into specific groups has two benefits: a group consists of game features with similar modus operandi (i.e., correlated and in the same context), and groups can serve as a basis for sharing research with other game genres.

We already mentioned (Section 2.1) that the literature search revealed 18 groups, which are formed independently of the specific game genre, and which we will use for grouping. These groups are: adaptation, assessment/rewards/scores, challenge, conflict, control, exploration, fantasy/location, interaction/interactivity (equipment), interaction (interpersonal/social), language/communication, motivation, mystery, pieces or players, progress and surprise, representation, rules/goals, safety and sensory stimuli. A detailed description about the meaning of each of the groups can be found in the tabular presentation in [62].

Table 2 presents the results after the completion of both steps, with references to the source of the compact description. Our goal was to have at least one game feature representative for each of the groups. If there was no game feature available in Table 1 for an empty group, we tried to locate the research work for that group by searching via Google Scholar (STEP 3) using different search strings (e.g., "impassable terrain" for a conflict group) in regard to the context of the group. The research

works found went through the procedure described in STEP 1, and a short game feature description was included in Table 1 and in the accordingly empty group in Table 2. (Observation: we noticed that many research works on game feature descriptions originated from the domain of player/opponent modeling, RTS replay analysis, game balancing and strategy selection/prediction.) One game feature can belong to more than one group. For some groups, we could not find or create any viable game feature description that could be measured by the game mechanics. Such groups remained empty but were still included in the Table. The reason: future RTS research could produce game features for currently empty groups.

Table 2. Game definition groups and their game feature representatives.

ID	Group	Short Game Feature Label
G_1	Adaptation	**GF3_DIFA** [1], GF10_DIFC
G_2	Assessment/Rewards/Scores	**GF5_AST** [1]
G_3	Challenge	GF3_DIFA, GF8_FANT, **GF10_DIFC** [1], GF12_INTE
G_4	Conflict	**GF6_SB** [1], GF9_PARI, GF10_DIFC, GF12_INTE, GF13_RG2
G_5	Control	GF2_EOBJ, GF9_PARI, GF10_DIFC, **GF11_GCMP** [1], GF12_INTE
G_6	Exploration	**GF7_EXPL** [1], GF9_PARI, GF12_INTE
G_7	Fantasy/Location	**GF8_FANT** [1]
G_8	Interaction/Interactivity (Equipment)	GF2_EOBJ, GF4_CONS, GF7_EXPL, **GF12_INTE** [1]
G_9	Interaction (Interpersonal/Social)	(empty—beyond the scope of this article [2])
G_{10}	Language/Communication	(empty)
G_{11}	Motivation	(empty)
G_{12}	Mystery	**GF9_PARI** [1]
G_{13}	Pieces or Players	GF1_RG, **GF2_EOBJ** [1], GF3_DIFA, GF4_CONS, GF5_AST, GF6_SB, GF7_EXPL, GF8_FANT, GF9_PARI, GF10_DIFC, GF11_GCMP, GF12_INTE, GF13_RG2
G_{14}	Progress and Surprise	GF1_RG, **GF4_CONS** [1], GF6_SB, GF7_EXPL, GF8_FANT, GF9_PARI, GF10_DIFC, GF11_GCMP, GF12_INTE, GF13_RG2
G_{15}	Representation	(empty)
G_{16}	Rules/goals	**GF1_RG** [1], GF2_EOBJ, GF4_CONS, GF7_EXPL, GF13_RG2
G_{17}	Safety	GF1_RG, **GF13_RG2** [1]
G_{18}	Sensory stimuli	(empty)

[1] Representative of the group used for the experiment. [2] The interaction (Interpersonal/Social) group was left empty, because it would require the interaction of multiple players (a single gameplaying agent modified for a playtesting agent supports only single player operations).

3.3. Classification of Feature Groups According to Their Correlation and Importance

As game features tend to be correlated, so do groups. One group can be, context wise, closely related to some groups but only loosely related to others. Additionally, some contexts are more important than others with regard to RTS gameplay.

Table 3 presents the classification of feature groups into three importance classes:

- The high-importance class contains groups that represent the essence of RTS gameplay (based on our understanding of the RTS game worlds and their aspects [63]);
- Groups that operate on a game mechanics level (e.g., Interaction/Interactivity (Equipment) group) or are not essential to the game (they could potentially be left out, e.g., Mystery group) are in the medium-importance class; and
- Groups that, in Table 2, did not have a feature representative (empty of features) were included in the low-importance class.

Table 3. Classification of feature groups based on their correlation and importance.

Class	Groups	Weight	Set
High importance	Adaptation, Challenge, Control, Pieces or Players, Progress and Surprise, Rules/goals, Safety	W1	$C_H = \{G_1, G_3, G_5, G_{13}, G_{14}, G_{16}, G_{17}\}$
Medium importance	Assessment/Rewards/Scores, Conflict, Exploration, Fantasy/Location, Interaction/Interactivity (Equipment), Motivation, Mystery	W2	$C_M = \{G_2, G_4, G_6, G_7, G_8, G_{12}\}$
Low importance	Interaction (Interpersonal/Social), Language/Communication, Representation, Sensory stimuli	W3	$C_L = \{G_9, G_{10}, G_{11}, G_{15}, G_{18}\}$

The importance level of each of the groups is represented by a class. Regarding the game worlds, we allow for the possibility of different reconfigurations of the groups inside the classes. We also included the weight and mathematical description of the set. Weight is a numerical value that is set by the user of the metric. It represents how much the groups belonging to the specific class will count towards the metric score.

3.4. Proposal of the Metric

In this subchapter, we explain our metric for summarizing agents' performance while they validate game features in an RTS game space. The metric calculates its score based on how many times the playtesting agent invalidated the game feature of a fixed number of repeats for a given scenario (the sum of validations and invalidations equals the number of scenario repeats).

If the playtesting agent during the execution of the scenario could not test the game feature, because it does not come into a situation, or it is not programmed to deal with the situation where validation can take place, then such a game feature is valid from this point of view. The number of successful validations is, therefore, omitted from the game score, since it is biased.

For a set of groups G_i, $1 \leq i \leq 18$, where each member of group G_i holds a set of Game features (GFs) ($GF_j \in G_i$, $0 \leq j$), and each GF_i holds a set of executable scenarios S ($S_k \in GF_i$, $1 \leq k$), the number of unsuccessful validations per scenario is defined by $numInvalid_{ijk}$, and the number of times the scenario is repeated is defined by numOfScenRep = n, $1 \leq n$, the following formulas apply:

$$\text{invalidPercPerScen}\ (i_{jk}, \text{numOfScenReps}) = \text{numInvalid}_{ijk}/\text{numOfScenRep} \tag{1}$$

$$\text{calcSetScore}\ (\text{set}, \text{numOfScenReps}) = \sum_{i\ \in\ \text{set}} \sum_{j\geq 0} \sum_{k\geq 1} \text{invalidPercPerScen}\ (ijk, \text{numOfScenRep}) \tag{2}$$

$$\begin{gathered}\text{agentPlaytestingScore} = \\ \text{W1} * \text{calcSetScore}\ (\{1, 3, 5, 13, 14, 16, 17\}, \text{numOfScenRep}) \\ + \text{W2} * \text{calcSetScore}\ (\{2, 4, 6, 7, 8, 12\}, \text{numOfScenRep}) \\ + \text{W3} * \text{calcSetScore}\ (\{9, 10, 11, 15, 18\}, \text{numOfScenRep})\end{gathered} \tag{3}$$

In Equation (1), the number of invalidations of a given group (index i), game feature (index j) and scenario (index k) is divided by the total number of scenario repeats. In Equation (2), the score is calculated for all the game features and scenarios that the set of groups holds. In Equation (3), the scores of the set of groups are multiplied by their respective weights.

4. Experiments and Results

In this chapter, we present the specifications of hardware and software used for the experimental environment, as well as the results of the experiments.

4.1. Experimental Environment

Hardware: The experiment was carried out on an i7-3770k CPU computer @ 3.50 (turbo: 3.9) GHz, 4 cores (note: during the experimentation, only one core was used, since agents do not implement the multi-core support) and 16 GB RAM.

Software: OS Windows 10 Pro and Java Development Kit 13.0.2. The experiment was set in the latest version of the microRTS environment, acquired from an online source at the time of preparing this article [64]. The microRTS environment comes pre-loaded with the following gameplaying agents: RandomAI, RandomAIBiased, MonteCarlo, IDRTMinimax, IDRTMinimaxRandomized, IDABCD, UCT, PuppetSearchMCTS, and NaiveMCTS. TiamatBot was acquired from the online source [65]. MixedBot (which includes TiamatBot source files but an improved version) was acquired from the online source [66] and was included in the microRTS environment. Every gameplaying agent is used in the experiment as it was acquired from the online source of original authors (i.e., no code or internal parameter was changed for experimental purposes).

Table 4 shows the hyper-parameters used for the validation of every game feature presented in Table 1.

Table 4. Hyper-parameters used in the experimentation.

Hyper-Parameter	Value
continuing	true
max_actions	100
max_playouts	−1
playout_time	100
max_depth	10
randomized_ab_repeats	10
max_cycles	3000
max_inactive_cycles	300

These hyper-parameters are pre-set within the microRTS environment. The only parameter that we changed was iterations, which we set to 50 (before it was set to 10). The standard UnitTypeTable was used where necessary. Note: to validate the GF9_PARI, we changed the environment from fully observable to partially observable.

The game feature descriptions presented in Table 1 were derived from related works and written independently of a specific game environment, i.e., they can be implemented in any RTS game engine. In Table 5, we present the same game features as those presented in Table 1, although the former are adapted to the microRTS environment and a specific scenario. All game features in Table 5 are written with the assumption that they are valid for the microRTS environment. If the playtesting agent actually manages to invalidate a game feature from the list, it will add to its metric score.

4.2. Adaptation of Gameplaying Agents as Playtesting Agents

To adapt a gameplaying agent to the playtesting task, we created a non-intrusive component. The component contains information about the scenario (map, position of units and the opponent) and controls the validation procedure by following the playtesting agents' progress (i.e., actions that it executes) and by accessing game environment information (e.g., current game state status). All the information is accessible through well-defined interfaces of the microRTS source code. One of the interface methods is the method that returns the best action for the given game state, and every gameplaying agent operating in the microRTS environment implements it.

Table 5. microRTS game feature scenario.

Short Game Feature Label	Experimental microRTS Game Feature Description	Map
GF1_RG	Worker collects at least 2 units of a resource in 2 trips.	basesWorkers8x8.xml (standard map, which comes with microRTS)
GF2_EOBJ	A light battle unit always hits with 2 points of damage.	melee4x4light2.xml (standard map)
GF3_DIFA	The opponent is aided by 5 more heavy battle units, resulting in the player losing every game.	basesWorkers8x8.xml (standard map with 5 heavy units added for the opponent)
GF4_CONS	If the player tries to, they must be able to create 1 barracks.	basesWorkers8x8.xml (standard map)
GF5_AST	The game score is calculated on the basis of raw features of the game state (no. of workers and no. of light, heavy and ranged units multiplied by their cost factors) and must represent the game state status correctly when presented to the player.	melee14x12Mixed18.xml (standard map)
GF6_SB	The player cannot destroy the enemy in a specific part of the map due to a wall.	basesWorkers12x12.xml (standard map with a wall placed in the middle of the map).
GF7_EXPL	If the player tries discovery, it must be able to create 1 light battle unit through the usage of game barracks.	basesWorkers8x8.xml (standard map)
GF8_FANT	The player is confronted with a special game unit (Super-Heavy battle unit with ten-times the armor of a normal-Heavy one), which cannot be destroyed with the given resources.	basesWorkers8x8 (standard map with Super-Heavy battle units added to help the opponent)
GF9_PARI	The player cannot operate in a partially observable environment, so it therefore cannot destroy the opponent in such an environment.	basesWorkers12x12.xml (standard map with a partially observable environment enabled)
GF10_DIFC	The player cannot destroy 2 barracks guarded with 3 heavy rushing units with access to 60 units of resources.	8x8_2barracks3rushingHeavy60res.xml (custom map)
GF11_GCMP	The player can destroy three barracks before the time runs out.	8x8_3barracks.xml (custom map)
GF12_INTE	If the player controlling four heavy battle units finds an enemy barracks in a large map (with obstacles and walls), the enemy barracks are always destroyed.	chambers32x32.xml (standard map with four heavy battle units and barracks added)
GF13_RG2	The worker is always destroyed when trying to gather resources.	8x8_workerDestroyed.xml (custom map with the base and resources on different parts of the map and four light battle units in the middle)

When the actions are executed in a game state, it cycles to the next one (i.e., actions change the inner state). During such cycles, our component tests if the Game Feature is valid or invalid. A Game Feature is invalid if the condition that is written in the validation procedure of the game feature in question is not fulfilled. The condition is tested against the information provided from the agent's executed action and the environment's current game state. The validation procedure checks the validity of the game feature, until either the maximum number of cycles is reached, or the game is over (i.e., one of the players has no more units left on the field).

For example, the game feature, GF8_FANT, is validated by checking if the resulting game state still holds this special unit after the agent has given an order to fire on it. If in any cycle the unit is destroyed, the game feature is invalid.

4.3. Playtesting Agents

The following gameplaying agents have been adapted as playtesting agents for the purposes of experimentation:

1. Basic (part of the microRTS package):
    - RandomAI: The choice of actions is completely random;
    - RandomBiasedAI: Based on RandomAI, but with a five times higher probability of choosing fighting or harvesting action over other actions; and
    - MonteCarlo: A standard Monte Carlo search algorithm.
2. Evolutionary Algorithm (online source):
    - TiamatBot (original): Uses an evolutionary procedure to derive action abstractions (conducted as a preprocessing step [67]). The generation of action abstractions can be cast as a problem of selecting a subset of pure strategies from a pool of options. It uses Stratified Strategy Selection (SSS) to plan in real time in the space defined by the action abstraction thus generated [68]. It outperformed the best performing methods in the 2017 microRTS competition [69] and is therefore considered as one of the current state-of-the-art gameplaying agents.
3. Tree-Based (part of the microRTS package):
    - IDRTMinimax: An iterative-deepening version of RTMinimax (minimax is defined here by time, not by agent moves) that uses available time to search in a tree as deeply as possible;
    - IDRTMinimaxRandomized: An agent that uses randomized alpha-beta (a better assessment for situations where players execute moves simultaneously);
    - IDABCD: Alpha-beta considering duration. It is a modified RTMinimax [70];
    - UCT: Standard UCT (with a UCB1 sampling policy);
    - PuppetSearchMCTS: An adversarial search framework based on scripts that can expose choice points to a look-ahead procedure. A Monte Carlo adversarial search tree was used to search over sequences of puppet moves. The input script into an agent's constructor was a basic configurable script that used a Unit Type Table [71].
    - NaiveMCTS: A Standard Monte Carlo search, but which uses naïve sampling [72]. Two variations of the same algorithm were used (which differ in their initial parameter settings): NaiveMCTS#A (max_Depth = 1, εl = 0.33, ε0 = 0.75) and NaiveMCTS#B (max_depth = 10, εl = 1.00, ε0 = 0.25).
4. Evolutionary and Tree-Based (online source):
    - MixedBot: This bot integrates three bots into a single agent. The TiamatBot (improved original) was used for strategy decisions, Capivara was used for tactical decisions [73], and MicroRTSbot [74] included a mechanism that could change the time allocated for two decision parts dynamically based on the number of close armies. MixedBot placed second in the 2019 microRTS (standard track) competition (first place went to the game bot that also uses offline/out-game learning [75]).

4.4. Results of the Playtesting Agents

For each of the playtesting agents, Table 6 shows how many times the group's game feature representative was validated or invalidated. Table 6 also shows what metric score they acquired. The weights for calculating the metric score were set as follows: W1 = 1, W2 = 0.5 and W3 = 0. W3 was set to 0, because the C_L class groups are devoid of features. Additionally, empty groups were omitted from the Table.

To allow for clearer results, we abbreviated the game feature representatives' labels, e.g., G_1 and its GF3_DIFA Game Feature representative, if validated 50 times and invalidated 0 times, was shortened to G1GF3(50, 0).

Table 6. Playtesting agent results for feature validations and their metric score.

Playtesting Agent	Groups and Game Features (Valid num./Invalid num.)	Metric Score
RandomAI	G1GF3(50, 0), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(50, 0), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	0
RandomBiasedAI	G1GF3(49, 1), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(28, 22), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	0.24
MonteCarlo	G1GF3(50, 0), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	0.5
TiamatBot	G1GF3(21, 29), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	1.08
IDRTMinimax	G1GF3(50, 0), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	0.5
IDRTMinimaxRandomized	G1GF3(50, 0), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	0.5
IDABCD	G1GF3(49, 1), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	0.52
UCT	G1GF3(24, 26), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	1.02
PuppetSearchMCTS	G1GF3(46, 4), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	0.58
NaiveMCTS#A	G1GF3(11, 39), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	1.28
NaiveMCTS#B	G1GF3(12, 38), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	1.26
MixedBot	G1GF3(34, 16), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	0.82

Table A1, which, due to its size, can be found in Appendix A, shows how the metric score changes for each of the playtesting agents and all combinations of the W1 to be decreased from 1 to 0.55 (with steps of 0.05) and those of W2 to be decreased from 0.50 to 0.05 (also with steps of 0.05). Note: the data

used for calculating the metric scores is the same as those presented in the second column of Table 6. RandomAI was omitted from Table A1, because its metric score is zero for all the combinations (it did not invalidate any of the features).

5. Discussion

During the experimentation phase, the microRTS game engine environment performed as expected (i.e., without visible or known bugs). Our presumption from the start of the experiment was that all of the Game Features were valid, yet the experiments showed that two of the Game Features were actually invalid (GF3 and GF8). A closer inspection of the GF3 results, specifically its invalidation number, revealed that not all of the playtesting agents caught the invalid game feature, and that some of them only invalidated it in a fraction of tries. Additionally, if the number of scenario repeats would be set to lower than fifty, it is possible that only the playtesting agents with a better performance would be successful in finding GF3 to be invalid.

GF3 was invalidated by eight playtesting agents, while GF8 was invalidated by all of them, with the only exception being the basic RandomAI. The difference in the number of playtesting agents that invalidated the game features, GF3 and GF8, successfully shows us that some game features are more sophisticated and require agents that intelligently explore and exploit the search space in question.

We discovered two important guidelines for validation testing:

1. Good agents' gameplaying performance is important, because it also reflects playtesting performance; and
2. With a greater number of scenario repeats comes a higher probability of game features being valid.

Our purpose was not to judge the existing gameplaying agents created by the research community based on the score they achieved. We did, however, use the invalid number part that they attained to calculate the metric score for metric testing purposes. The results were encouraging. The state-of-the art evolutionary and tree-based agents were good performers, not just for gameplaying, but also for playtesting. The line between basic agents (e.g., G1GF3(50, 0)) and advanced ones (e.g., G1GF3(21, 29)) can also be clearly seen. We did not measure the average time for an agent to complete a scenario, but during playtesting, we noticed that agents that were either basic (e.g., RandomAI) or very good performers (e.g., NaiveMCTS) completed the validations in the fastest amount of time. We believe that this resulted from decisions being made quickly (either bad or good).

At this point, we can also provide answers to the research questions presented in the Introduction. RQ1: the adaptation of a gameplaying agent as a playtesting agent is straightforward, provided that the game engine follows good software design techniques (components, interfaces, etc.). In our estimation, this is very important, because it allows for research discoveries in the gameplaying domain to be transferred to the playtesting domain and probably also for higher adaptation rates of such discoveries for commercial use. RQ2: In comparing different playtesting agents, our metric relies on the groups presented in Table 2. The groups belong to different classes (Table 3), each with their own weights. Additional information for comparisons can also be found based on the calibration of these weights. For that purpose, Table A1 was created in Appendix A, which shows how the metric score changes in relation to the changes of the weights. In this way, we can give importance to a specific set of groups and achieve a greater differentiation between the playtesting agents covering them. RQ3: playtesting agents are evaluated through game feature definitions using the created metric. The most beneficial Game Feature definitions are the ones that belong to the groups that are in the high-importance class, shown in Table 3. RQ4: evolutionary and non-evolutionary approaches in the state-of-the-art segment both performed well, and their playtesting abilities are high. No major differences were detected for the game features and scenarios tested. RQ5: the validity of the game feature was defined, with the condition of the validation procedure inside the component used for the adaptation of the gameplaying agents.

6. Conclusions

The experiments provide encouraging results, and we confirmed our belief that playtesting with agents is important and worthy of further research. Playtesting agents can play in the same scenario repeatedly and with good results, while repetitive play (e.g., playing the same scenario fifty or more times) is probably tiresome for human players, who are therefore more prone to making errors. We also confirmed that our novel metric performed as expected, because the metric scores revealed a certain consistency when traversing from basic to state-of-the art playtesting agents. To the knowledge of the authors, such a metric (i.e., one that would evaluate playtesting game agents based on their game feature performance) does not yet exist. The creation of it is necessary to establish common ground for the research conducted in the domain of game features and in the domain of playtesting agents.

Through a series of experiments, we were also interested in how different evolutionary-based playtesting agents explored the search space. The valuable information obtained in our experiments will serve us as a steppingstone in the development of new playtesting agents that are based on modern Evolutionary Algorithms, as well as Swarm Intelligence algorithms.

Author Contributions: Conceptualization, D.N., D.V. and I.F.J.; methodology, D.N. and I.F.J.; software, D.N. and J.D.; validation, D.N., I.F.J. and D.V.; formal analysis, D.N. and I.F.J.; investigation, D.N.; resources, D.N., I.F.J. and J.D.; data curation, J.D.; writing—original draft preparation, D.N. and I.F.J.; writing—review and editing, D.N., I.F.J. and J.D.; visualization, J.D.; supervision, I.F.J.; project administration, J.D. and D.V. All authors have read and agreed to the published version of the manuscript.

Funding: The authors acknowledge the financial support from the Slovenian Research Agency (Research Core Funding No. P2-0057).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Metric scores with variable weights for all the playtesting agents.

Playtesting Agent	Metric Scores										
		1	**0.95**	**0.9**	**0.85**	**0.8**	**0.75**	**0.7**	**0.65**	**0.6**	**0.55**
RandomBiasedAI	**0.5**	0.24	0.239	0.238	0.237	0.236	0.235	0.234	0.233	0.232	0.231
	0.45	0.218	0.217	0.216	0.215	0.214	0.213	0.212	0.211	0.21	0.209
	0.4	0.196	0.195	0.194	0.193	0.192	0.191	0.19	0.189	0.188	0.187
	0.35	0.174	0.173	0.172	0.171	0.17	0.169	0.168	0.167	0.166	0.165
	0.3	0.152	0.151	0.15	0.149	0.148	0.147	0.146	0.145	0.144	0.143
	0.25	0.13	0.129	0.128	0.127	0.126	0.125	0.124	0.123	0.122	0.121
	0.2	0.108	0.107	0.106	0.105	0.104	0.103	0.102	0.101	0.1	0.099
	0.15	0.086	0.085	0.084	0.083	0.082	0.081	0.08	0.079	0.078	0.077
	0.1	0.064	0.063	0.062	0.061	0.06	0.059	0.058	0.057	0.056	0.055
	0.05	0.042	0.041	0.04	0.039	0.038	0.037	0.036	0.035	0.034	0.033
		1	**0.95**	**0.9**	**0.85**	**0.8**	**0.75**	**0.7**	**0.65**	**0.6**	**0.55**
MonteCarlo	**0.5**	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35
	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15
	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05

Table A1. *Cont.*

Playtesting Agent		Metric Scores									
		1	**0.95**	**0.9**	**0.85**	**0.8**	**0.75**	**0.7**	**0.65**	**0.6**	**0.55**
TiamatBot	**0.5**	1.08	1.051	1.022	0.993	0.964	0.935	0.906	0.877	0.848	0.819
	0.45	1.03	1.001	0.972	0.943	0.914	0.885	0.856	0.827	0.798	0.769
	0.4	0.98	0.951	0.922	0.893	0.864	0.835	0.806	0.777	0.748	0.719
	0.35	0.93	0.901	0.872	0.843	0.814	0.785	0.756	0.727	0.698	0.669
	0.3	0.88	0.851	0.822	0.793	0.764	0.735	0.706	0.677	0.648	0.619
	0.25	0.83	0.801	0.772	0.743	0.714	0.685	0.656	0.627	0.598	0.569
	0.2	0.78	0.751	0.722	0.693	0.664	0.635	0.606	0.577	0.548	0.519
	0.15	0.73	0.701	0.672	0.643	0.614	0.585	0.556	0.527	0.498	0.469
	0.1	0.68	0.651	0.622	0.593	0.564	0.535	0.506	0.477	0.448	0.419
	0.05	0.63	0.601	0.572	0.543	0.514	0.485	0.456	0.427	0.398	0.369
		1	**0.95**	**0.9**	**0.85**	**0.8**	**0.75**	**0.7**	**0.65**	**0.6**	**0.55**
IDRTMinimax	**0.5**	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35
	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15
	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
		1	**0.95**	**0.9**	**0.85**	**0.8**	**0.75**	**0.7**	**0.65**	**0.6**	**0.55**
IDRTMinimaxRandomized	**0.5**	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35
	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15
	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
		1	**0.95**	**0.9**	**0.85**	**0.8**	**0.75**	**0.7**	**0.65**	**0.6**	**0.55**
IDABCD	**0.5**	0.52	0.519	0.518	0.517	0.516	0.515	0.514	0.513	0.512	0.511
	0.45	0.47	0.469	0.468	0.467	0.466	0.465	0.464	0.463	0.462	0.461
	0.4	0.42	0.419	0.418	0.417	0.416	0.415	0.414	0.413	0.412	0.411
	0.35	0.37	0.369	0.368	0.367	0.366	0.365	0.364	0.363	0.362	0.361
	0.3	0.32	0.319	0.318	0.317	0.316	0.315	0.314	0.313	0.312	0.311
	0.25	0.27	0.269	0.268	0.267	0.266	0.265	0.264	0.263	0.262	0.261
	0.2	0.22	0.219	0.218	0.217	0.216	0.215	0.214	0.213	0.212	0.211
	0.15	0.17	0.169	0.168	0.167	0.166	0.165	0.164	0.163	0.162	0.161
	0.1	0.12	0.119	0.118	0.117	0.116	0.115	0.114	0.113	0.112	0.111
	0.05	0.07	0.069	0.068	0.067	0.066	0.065	0.064	0.063	0.062	0.061
		1	**0.95**	**0.9**	**0.85**	**0.8**	**0.75**	**0.7**	**0.65**	**0.6**	**0.55**
UCT	**0.5**	1.02	0.994	0.968	0.942	0.916	0.89	0.864	0.838	0.812	0.786
	0.45	0.97	0.944	0.918	0.892	0.866	0.84	0.814	0.788	0.762	0.736
	0.4	0.92	0.894	0.868	0.842	0.816	0.79	0.764	0.738	0.712	0.686
	0.35	0.87	0.844	0.818	0.792	0.766	0.74	0.714	0.688	0.662	0.636
	0.3	0.82	0.794	0.768	0.742	0.716	0.69	0.664	0.638	0.612	0.586
	0.25	0.77	0.744	0.718	0.692	0.666	0.64	0.614	0.588	0.562	0.536
	0.2	0.72	0.694	0.668	0.642	0.616	0.59	0.564	0.538	0.512	0.486
	0.15	0.67	0.644	0.618	0.592	0.566	0.54	0.514	0.488	0.462	0.436
	0.1	0.62	0.594	0.568	0.542	0.516	0.49	0.464	0.438	0.412	0.386
	0.05	0.57	0.544	0.518	0.492	0.466	0.44	0.414	0.388	0.362	0.336

Table A1. *Cont.*

Playtesting Agent		Metric Scores									
PuppetSearchMCTS		**1**	**0.95**	**0.9**	**0.85**	**0.8**	**0.75**	**0.7**	**0.65**	**0.6**	**0.55**
	0.5	0.58	0.576	0.572	0.568	0.564	0.56	0.556	0.552	0.548	0.544
	0.45	0.53	0.526	0.522	0.518	0.514	0.51	0.506	0.502	0.498	0.494
	0.4	0.48	0.476	0.472	0.468	0.464	0.46	0.456	0.452	0.448	0.444
	0.35	0.43	0.426	0.422	0.418	0.414	0.41	0.406	0.402	0.398	0.394
	0.3	0.38	0.376	0.372	0.368	0.364	0.36	0.356	0.352	0.348	0.344
	0.25	0.33	0.326	0.322	0.318	0.314	0.31	0.306	0.302	0.298	0.294
	0.2	0.28	0.276	0.272	0.268	0.264	0.26	0.256	0.252	0.248	0.244
	0.15	0.23	0.226	0.222	0.218	0.214	0.21	0.206	0.202	0.198	0.194
	0.1	0.18	0.176	0.172	0.168	0.164	0.16	0.156	0.152	0.148	0.144
	0.05	0.13	0.126	0.122	0.118	0.114	0.11	0.106	0.102	0.098	0.094
NaiveMCTS#A		**1**	**0.95**	**0.9**	**0.85**	**0.8**	**0.75**	**0.7**	**0.65**	**0.6**	**0.55**
	0.5	1.28	1.241	1.202	1.163	1.124	1.085	1.046	1.007	0.968	0.929
	0.45	1.23	1.191	1.152	1.113	1.074	1.035	0.996	0.957	0.918	0.879
	0.4	1.18	1.141	1.102	1.063	1.024	0.985	0.946	0.907	0.868	0.829
	0.35	1.13	1.091	1.052	1.013	0.974	0.935	0.896	0.857	0.818	0.779
	0.3	1.08	1.041	1.002	0.963	0.924	0.885	0.846	0.807	0.768	0.729
	0.25	1.03	0.991	0.952	0.913	0.874	0.835	0.796	0.757	0.718	0.679
	0.2	0.98	0.941	0.902	0.863	0.824	0.785	0.746	0.707	0.668	0.629
	0.15	0.93	0.891	0.852	0.813	0.774	0.735	0.696	0.657	0.618	0.579
	0.1	0.88	0.841	0.802	0.763	0.724	0.685	0.646	0.607	0.568	0.529
	0.05	0.83	0.791	0.752	0.713	0.674	0.635	0.596	0.557	0.518	0.479
NaiveMCTS#B		**1**	**0.95**	**0.9**	**0.85**	**0.8**	**0.75**	**0.7**	**0.65**	**0.6**	**0.55**
	0.5	1.26	1.222	1.184	1.146	1.108	1.07	1.032	0.994	0.956	0.918
	0.45	1.21	1.172	1.134	1.096	1.058	1.02	0.982	0.944	0.906	0.868
	0.4	1.16	1.122	1.084	1.046	1.008	0.97	0.932	0.894	0.856	0.818
	0.35	1.11	1.072	1.034	0.996	0.958	0.92	0.882	0.844	0.806	0.768
	0.3	1.06	1.022	0.984	0.946	0.908	0.87	0.832	0.794	0.756	0.718
	0.25	1.01	0.972	0.934	0.896	0.858	0.82	0.782	0.744	0.706	0.668
	0.2	0.96	0.922	0.884	0.846	0.808	0.77	0.732	0.694	0.656	0.618
	0.15	0.91	0.872	0.834	0.796	0.758	0.72	0.682	0.644	0.606	0.568
	0.1	0.86	0.822	0.784	0.746	0.708	0.67	0.632	0.594	0.556	0.518
	0.05	0.81	0.772	0.734	0.696	0.658	0.62	0.582	0.544	0.506	0.468
MixedBot		**1**	**0.95**	**0.9**	**0.85**	**0.8**	**0.75**	**0.7**	**0.65**	**0.6**	**0.55**
	0.5	0.82	0.84	0.788	0.772	0.756	0.74	0.724	0.708	0.692	0.676
	0.45	0.77	0.754	0.738	0.722	0.706	0.69	0.674	0.658	0.642	0.626
	0.4	0.72	0.704	0.688	0.672	0.656	0.64	0.624	0.608	0.592	0.576
	0.35	0.67	0.654	0.638	0.622	0.606	0.59	0.574	0.558	0.542	0.526
	0.3	0.62	0.604	0.588	0.572	0.556	0.54	0.524	0.508	0.492	0.476
	0.25	0.57	0.554	0.538	0.522	0.506	0.49	0.474	0.458	0.442	0.426
	0.2	0.52	0.504	0.488	0.472	0.456	0.44	0.424	0.408	0.392	0.376
	0.15	0.47	0.454	0.438	0.422	0.406	0.39	0.374	0.358	0.342	0.326
	0.1	0.42	0.404	0.388	0.372	0.356	0.34	0.324	0.308	0.292	0.276
	0.05	0.37	0.354	0.338	0.322	0.306	0.29	0.274	0.258	0.242	0.226

References

1. Balla, R.K.; Fern, A. UCT for Tactical Assault Planning in Real-Time Strategy Games. In Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence, Pasadena, CA, USA, 14–17 July 2009; AAAI Press: Menlo Park, CA, USA, 2009; pp. 40–45.
2. Buro, M. Real-Time Strategy Games: A New AI Research Challenge. In Proceedings of the IJCAI, Acapulco, Mexico, 9–15 August 2003; Morgan Kaufmann: Burlington, MA, USA, 2003; pp. 1534–1535.
3. Shafi, K.; Abbass, H.A. A Survey of Learning Classifier Systems in Games. *IEEE Comput. Intell. Mag.* **2017**, *12*, 42–55. [CrossRef]
4. Synnaeve, G.; Bessiere, P. Multi-scale Bayesian modeling for RTS games: An application to StarCraft AI. *IEEE Trans. Comput. Intell. AI Games* **2015**, *8*, 338–350. [CrossRef]
5. Usunier, N.; Synnaeve, G.; Lin, Z.; Chintala, S. Episodic Exploration for Deep Deterministic Policies: An Application to StarCraft Micromanagement Tasks. *arXiv* **2016**, arXiv:1609.02993.

6. Isaksen, A.; Gopstein, D.; Nealen, A. Exploring Game Space Using Survival Analysis. In Proceedings of the FDG, Pacific Grove, CA, USA, 22–25 June 2015.
7. Gottlob, G.; Greco, G.; Scarcello, F. Pure Nash Equilibria: Hard and Easy Games. *JAIR* **2005**, *24*, 357–406. [CrossRef]
8. Eiben, A.E.; Smith, J.E. *Introduction to Evolutionary Computing*; Eiben, A.E., Ed.; Springer: Berlin, Germany, 2003; Volume 53, p. 18.
9. Fister, I., Jr.; Yang, X.S.; Fister, I.; Brest, J.; Fister, D. A brief review of nature-inspired algorithms for optimization. *arXiv* **2013**, arXiv:1307.4186.
10. Yang, X.S. *Nature-Inspired Metaheuristic Algorithms*; Luniver Press: Beckington, UK, 2010.
11. Yang, X.S. *Nature-Inspired Optimization Algorithms*; Elsevier: London/Waltham, UK, 2014.
12. Biswas, A.; Mishra, K.; Tiwari, S.; Misra, A. Physics-Inspired Optimization Algorithms: A Survey. *J. Optim.* **2013**. [CrossRef]
13. Del Ser, J.; Osaba, E.; Molina, D.; Yang, X.S.; Salcedo-Sanz, S.; Camacho, D.; Das, S.; Suganthan, P.; Coello, C.; Herrera, F. Bio-inspired computation: Where we stand and what's next. *SWEVO* **2019**, *48*. [CrossRef]
14. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
15. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
16. Goldberg, D.E. Genetic algorithms in search. In *Optimization, and Machine Learning*; Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 1989.
17. Wang, G.G.; Deb, S.; Cui, Z. Monarch Butterfly Optimization. *Neural Comput. Appl.* **2015**, *31*, 1995–2014. [CrossRef]
18. Jin, N.; Rahmat-Samii, Y. Advances in Particle Swarm Optimization for Antenna Designs: Real-Number, Binary, Single-Objective and Multiobjective Implementations. *IEEE Trans. Antennas Propag.* **2007**, *55*, 556–567. [CrossRef]
19. Santucci, V.; Milani, A.; Caraffini, F. An Optimisation-Driven Prediction Method for Automated Diagnosis and Prognosis. *Mathematics* **2019**, *7*, 1051. [CrossRef]
20. Yeoh, J.M.; Caraffini, F.; Homapour, E.; Santucci, V.; Milani, A. A Clustering System for Dynamic Data Streams Based on Metaheuristic Optimisation. *Mathematics* **2019**, *7*, 1229. [CrossRef]
21. Hendrikx, M.; Meijer, S.; Van Der Velden, J.; Iosup, A. Procedural content generation for games: A survey. *ACM Trans. Multimed. Comput. Commun. Appl.* **2013**, *9*, 1–22. [CrossRef]
22. Wilson, D.G.; Cussat-Blanc, S.; Luga, H.; Miller, J.F. Evolving simple programs for playing Atari games. *Proc. Genet. Evol. Comput. Conf.* **2018**, 229–236. [CrossRef]
23. Ponticorvo, M.; Rega, A.; Di Ferdinando, A.; Marocco, D.; Miglino, O. Approaches to Embed Bio-inspired Computational Algorithms in Educational and Serious Games. In Proceedings of the CAID@ IJCAI, Melbourne, Australia, May 2017.
24. Woźniak, M.; Połap, D.; Napoli, C.; Tramontana, E. Application of Bio-Inspired Methods in Distributed Gaming Systems. *ITC* **2017**, *46*. [CrossRef]
25. Boskovic, B.; Greiner, S.; Brest, J.; Zumer, V. A differential evolution for the tuning of a chess evaluation function. In Proceedings of the 2006 IEEE International Conference on Evolutionary Computation, Vancouver, BC, Canada, 16–21 July 2006; pp. 1851–1856.
26. Diaz, G.; Iglesias, A. Evolutionary Behavioral Design of Non-Player Characters in a FPS Video Game through Particle Swarm Optimization. In Proceedings of the 13th International Conference on SKIMA, Island of Ulkulhas, Ulkulhas, Maldives, 26–28 August 2019; pp. 1–8. [CrossRef]
27. Kuhlmann, G.; Stone, P. Automatic Heuristic Construction in a Complete General Game Player. *AAAI Conf.* **2006**, *6*, 1456–1462.
28. Joppen, T.; Strubig, T.; Furnkranz, J. Ordinal Bucketing for Game Trees using Dynamic Quantile Approximation. In Proceedings of the IEEE CoG, London, UK, 20–23 August 2019; pp. 1–8. [CrossRef]
29. Borovikov, I.; Zhao, Y.; Beirami, A.; Harder, J.; Kolen, J.; Pestrak, J.; Pinto, J.; Pourabolghasem, R.; Chaput, H.; Sardari, M.; et al. Winning isn't everything: Training agents to playtest modern games. In Proceedings of the AAAI Workshop on Reinforcement Learning in Games, Honolulu, HI, USA, 27 January–1 February 2019.

30. Naves, T.; Lopes, C. One Approach to Determine Goals in RTS Games Using Maximization of Resource Production with Local Search and Scheduling. In Proceedings of the ICTAI, Vietri sul Mare, Italy, 9–11 November 2015; pp. 469–477. [CrossRef]
31. Bosc, G.; Tan, P.; Boulicaut, J.F.; Raïssi, C.; Kaytoue, M. A Pattern Mining Approach to Study Strategy Balance in RTS Games. *IEEE T-CIAIG* **2015**, *9*, 123–132. [CrossRef]
32. Uriarte, A.; Ontañón, S. Combat Models for RTS Games. *IEEE TOG* **2018**, *10*, 29–41. [CrossRef]
33. Rogers, K.; Skabar, A. A Micromanagement Task Allocation System for Real-Time Strategy Games. *IEEE TCIAIG* **2014**, *6*, 67–77. [CrossRef]
34. Kawase, K.; Thawonmas, R. Scout of the route of entry into the enemy camp in StarCraft with potential field. In Proceedings of the GCCE, Tokyo, Japan, 1–4 October 2013; pp. 318–319. [CrossRef]
35. Cunha, R.; Chaimowicz, L. An Artificial Intelligence System to Help the Player of Real-Time Strategy Games. In Proceedings of the SBGames, Florianopolis, Brazil, 8–10 November 2010; pp. 71–81. [CrossRef]
36. Ontañón, S.; Synnaeve, G.; Uriarte, A.; Richoux, F.; Churchill, D.; Preuss, M. A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. *IEEE T-CIAIG* **2013**, *5*, 293–311. [CrossRef]
37. Zhao, Y.; Borovikov, I.; Beirami, A.; Rupert, J.; Somers, C.; Harder, J.; De Mesentier Silva, F.; Kolen, J.; Pinto, J.; Pourabolghasem, R.; et al. Winning Isn't Everything: Enhancing Game Development with Intelligent Agents. In Proceedings of the AAAI Workshop on Reinforcement Learning in Games, Honolulu, HI, USA, 27 January–1 February 2019.
38. Guerrero-Romero, C.; Lucas, S.; Perez Liebana, D. Using a Team of General AI Algorithms to Assist Game Design and Testing. In Proceedings of the IEEE Conference on CIG, Maastricht, The Netherlands, 14–17 August 2018; pp. 1–8. [CrossRef]
39. Jaffe, A.B. Understanding Game Balance with Quantitative Methods. Ph.D. Thesis, University of Washington, Washington, DC, USA, 2013.
40. Risi, S.; Preuss, M. From Chess and Atari to StarCraft and Beyond: How Game AI is Driving the World of AI. *KI-Künstliche Intell.* **2020**, *34*, 1–11. [CrossRef]
41. Perrotta, C.; Bailey, C.; Ryder, J.; Haggis-Burridge, M.; Persico, D. Games as (Not) Culture: A Critical Policy Analysis of the Economic Agenda of Horizon 2020. *Games Cult.* **2019**. [CrossRef]
42. Salazar, M.G.; Mitre, H.A.; Olalde, C.L.; Sánchez, J.L.G. Proposal of Game Design Document from software engineering requirements perspective. In Proceedings of the Conference on CGAMES, Louisville, KY, USA, 30 July–1 August 2012; pp. 81–85.
43. Holmgård, C.; Green, M.C.; Liapis, A.; Togelius, J. Automated Playtesting with Procedural Personas through MCTS with Evolved Heuristics. *IEEE Trans. Games* **2018**, *11*, 352–362. [CrossRef]
44. Chaslot, G.; Bakkes, S.; Szita, I.; Spronck, P. Monte-Carlo Tree Search: A New Framework for Game AI. In Proceedings of the AAAI Conference on AIIDE, Palo Alto, CA, USA, 22–24 October 2008.
45. Heintz, S.; Law, E. Digital Educational Games: Methodologies for Evaluating the Impact of Game Type. *ACM Trans. Comput. Hum. Interact.* **2018**, *25*, 1–47. [CrossRef]
46. Walfisz, M.; Zackariasson, P.; Wilson, T. Real-Time Strategy: Evolutionary Game Development. *Bus. Horiz.* **2006**, *49*, 487–498. [CrossRef]
47. Sicart, M. Defining Game Mechanics. *Int. J. Comput. Game Res.* **2008**, *8*.
48. Wilson, K.; Bedwell, W.; Lazzara, E.; Salas, E.; Burke, S.; Estock, J.; Orvis, K.; Conkey, C. Relationships Between Game Attributes and Learning Outcomes: Review and Research Proposals. *Simul. Gaming* **2008**, *40*, 217–266. [CrossRef]
49. Erickson, G.; Buro, M. Global state evaluation in StarCraft. In Proceedings of the AAAI Conference on AIIDE, Raleigh, NC, USA, 3–7 October 2014; pp. 112–118.
50. Ludwig, J.; Farley, A. Examining Extended Dynamic Scripting in a Tactical Game Framework. In Proceedings of the Conference on AIIDE, Palo Alto, CA, USA, 14–16 October 2009.
51. Aly, M.; Aref, M.; Hassan, M. Dimensions-based classifier for strategy classification of opponent models in real-time strategy games. In Proceedings of the IEEE Seventh ICICIS, Cairo, Egypt, 12–14 December 2015; pp. 442–446.
52. Bangay, S.; Makin, O. Generating an attribute space for analyzing balance in single unit RTS game combat. In Proceedings of the IEEE Conference on CIG, Dortmund, Germany, 26–29 August 2014; pp. 1–8. [CrossRef]
53. Cho, H.; Park, H.; Kim, C.Y.; Kim, K.J. Investigation of the Effect of "Fog of War" in the Prediction of StarCraft Strategy Using Machine Learning. *Comput. Entertain.* **2017**, *14*, 1–16. [CrossRef]

54. Mishra, K.; Ontañón, S.; Ram, A. Situation Assessment for Plan Retrieval in Real-Time Strategy Games. In *Advances in Case-Based Reasoning, ECCBR 2008*; Althoff, K.D., Bergmann, R., Minor, M., Hanft, A., Eds.; Lecture Notes in Computer Science; Springer: Berlin, Germany, 2008; Volume 5239, pp. 355–369.
55. Togelius, J.; Preuss, M.; Hochstrate, N.; Wessing, S.; Hagelbäck, J.; Yannakakis, G. Multiobjective exploration of the StarCraft map space. *IEEE Conf. CIG* **2010**, *1*, 265–272. [CrossRef]
56. Lin, M.; Wang, T.; Li, X.; Liu, J.; Wang, Y.; Zhu, Y.; Wang, W. An Uncertainty-Incorporated Approach to Predict the Winner in StarCraft II Using Neural Processes. *IEEE Access* **2019**, *7*, 101609–101619. [CrossRef]
57. Tong, C.; On, C.; Teo, J.; Chua, B.L. Automatic generation of real time strategy tournament units using differential evolution. In Proceedings of the IEEE CSUDET, Semenyih, Malaysia, 20–21 October 2011; pp. 101–106. [CrossRef]
58. Long, M. Radio General: A Real-Time Strategy Game Where You Cannot See Your Units. In Proceedings of the Annual Symposium on CHI PLAY, Melbourne, Australia, 28–31 October 2018; pp. 345–351. [CrossRef]
59. Li, Y.; Li, Y.; Zhai, J.; Shiu, S. RTS game strategy evaluation using extreme learning machine. *Soft Comput.* **2012**. [CrossRef]
60. Si, C.; Pisan, Y.; Tan, C.T. A Scouting Strategy for Real-Time Strategy Games. *Conf. Interact. Entertain.* **2014**, 1–8. [CrossRef]
61. McCoy, J.; Mateas, M. An Integrated Agent for Playing Real-Time Strategy Games. *AAAI Conf. AI* **2008**, *8*, 1313–1318.
62. DeRouin-Jessen, R. Game on: The Impact of Game Features in Computer-Based Training. Ph.D. Thesis, University of Central Florida, Orlando, FL, USA, 2008.
63. Novak, D.; Čep, A.; Verber, D. Classification of modern real-time strategy game worlds. *GSTF J. Comput.* **2018**, *6*. [CrossRef]
64. Microrts. Available online: https://github.com/santiontanon/microrts (accessed on 20 March 2020).
65. TiamatBot. Available online: https://github.com/jr9Hernandez/TiamatBot (accessed on 20 March 2020).
66. MixedBotmRTS. Available online: https://github.com/AmoyZhp/MixedBotmRTS (accessed on 20 March 2020).
67. Evolutionary Action-Abstractions. Available online: https://github.com/julianmarino/evolutionary-action-abstractions (accessed on 15 April 2020).
68. Mariño, J.; De Oliveira Moraes Filho, R.; Toledo, C.; Lelis, L. Evolving Action Abstractions for Real-Time Planning in Extensive-Form Games. *AAAI Conf. AI* **2019**, *33*, 2330–2337. [CrossRef]
69. Ontanon, S.; Barriga, N.A.; Silva, C.; De Oliveira Moraes Filho, R.; Lelis, L. The First microRTS Artificial Intelligence Competition. *AI Mag.* **2018**, *39*, 75. [CrossRef]
70. Churchill, D.; Saffidine, A.; Buro, M. Fast Heuristic Search for RTS Game Combat Scenarios. In Proceedings of the AAAI Conference on AIIDE, Stanford, CA, USA, 8–12 October 2012.
71. Barriga, N.A.; Stanescu, M.; Buro, M. Game Tree Search Based on Non-Deterministic Action Scripts in Real-Time Strategy Games. *IEEE TCIAIG* **2017**. [CrossRef]
72. Ontanon, S. The combinatorial Multi-armed Bandit problem and its application to real-time strategy games. In Proceedings of the Conference on AIIDE, Boston, MA, USA, 14–18 October 2013; pp. 58–64.
73. De Oliveira Moraes Filho, R.; Mariño, J.; Lelis, L.; Nascimento, M. Action Abstractions for Combinatorial Multi-Armed Bandit Tree Search. In Proceedings of the Conference on AIIDE, Edmonton, AB, Canada, 13–17 November 2018.
74. Barriga, N.A.; Stanescu, M.; Buro, M. Combining Strategic Learning and Tactical Search in Real-Time Strategy Games. In Proceedings of the AAAI Conference on AIIDE, Snowbird, UT, USA, 5–9 October 2017.
75. Stanley, K.; Bryant, B.; Miikkulainen, R. Real-Time Neuroevolution in the NERO Video Game. *IEEE TEVC* **2005**, *9*, 653–668. [CrossRef]

mathematics

MDPI

Article

Ranking Multi-Metric Scientific Achievements Using a Concept of Pareto Optimality

Shahryar Rahnamayan [1], Sedigheh Mahdavi [1], Kalyanmoy Deb [2] and Azam Asilian Bidgoli [1,*]

[1] Nature Inspired Computational Intelligence (NICI) Lab, Department of Electrical, Computer, and Software Engineering, Ontario Tech University, Oshawa, ON L1G 0C5, Canada; shahryar.rahnamayan@uoit.ca (S.R.); sedigheh.mahdavi@uoit.ca (S.M.)

[2] Department of Electrical and Computer Engineering, Michigan State University (MSU), East Lansing, MI 48824, USA; kdeb@egr.msu.edu

* Correspondence: azam.asilianbidgoli@uoit.ca

Received: 18 April 2020; Accepted: 2 June 2020; Published: 11 June 2020

Abstract: The ranking of multi-metric scientific achievements is a challenging task. For example, the scientific ranking of researchers utilizes two major types of indicators; namely, number of publications and citations. In fact, they focus on how to select proper indicators, considering only one indicator or combination of them. The majority of ranking methods combine several indicators, but these methods are faced with a challenging concern—the assignment of suitable/optimal weights to the targeted indicators. Pareto optimality is defined as a measure of efficiency in the multi-objective optimization which seeks the optimal solutions by considering multiple criteria/objectives simultaneously. The performance of the basic Pareto dominance depth ranking strategy decreases by increasing the number of criteria (generally speaking, when it is more than three criteria). In this paper, a new, modified Pareto dominance depth ranking strategy is proposed which uses some dominance metrics obtained from the basic Pareto dominance depth ranking and some sorted statistical metrics to rank the scientific achievements. It attempts to find the clusters of compared data by using all of indicators simultaneously. Furthermore, we apply the proposed method to address the multi-source ranking resolution problem which is very common these days; for example, there are several world-wide institutions which rank the world's universities every year, but their rankings are not consistent. As our case studies, the proposed method was used to rank several scientific datasets (i.e., researchers, universities, and countries) for proof of concept.

Keywords: Pareto optimality; *h*-index; ranking; dominance; Pareto-front; multi-indicators; multi-metric; multi-resources; citation; universities ranking

1. Introduction

Nowadays, ranking of scientific impacts is a crucial task and it is a focus of research communities, universities, and governmental funding agencies. In this ranking, the target entities can be researchers, universities, countries, journals, or conferences. Performance analysis and benchmarking of scientific achievement has a variety of substantial purposes. At the researcher level, the research's impact is an important measure to define the main rules of academic institutions and universities on determination of funding, hiring, and promotions [1–3]. From the university's view point, university rankings are considered as a source of strategic information for governments, funding agencies, and the media in order to compare universities; then students and their parents use university rankings as a selection criterion [4]. As the assessment of scientific achievement has gained a great deal of attention for various interested groups, such as students, parents, institutions, academicians, policy makers, political leaders, donors/funding agencies, and news media; several assessment methods have been

developed in the field of bibliometry and scientometrics through the utilization of mathematical and/or statistical methods [1].

In order to measure a researcher's performance, many indicators have been proposed which can also be utilized in other scientific areas. Traditional research indicators include the numbers of publications and citations, the average number of citations per paper, and the average number of citations per year [5]. In 2005, Hirsch [6] proposed a new indicator, called *h*-index, which revolutionized scientometrics (informetrics). The original definition of the *h*-index indicator is that, "A scientist has the index h if h of his/her N_p papers have at least received h citations each, and the other $N_p - h$ papers have no more than h citations each." Later, other indicators were proposed to enhance the *h*-index. Additionally, *h*-index was defined for other scientific aggregation levels [7]. Ranking methods at researcher level tend to use only one indicator (*h*-index or its improved versions), but at other aggregation scientific levels they prefer to have a more comprehensive set of indicators. Research works in the scientometrics can be divided into the following two main categories: the first category includes methods which focus on introducing new indicators to enhance the performances of assessment metrics, and in the second category, methods attempt to develop enhanced ranking methods for obtaining ranks by using several various indicators.

There are various kinds of ranking methods; first, methods which focus only on one indicator; and second, methods which combine several of them. Considering only a specific indicator makes differences among the quality assessments of research outcomes very hard to be revealed. On the other hand, there are a few challenges for considering several indicators simultaneously. For instance, the method needs to find the proper weights for combining the indicators and also an efficient merging strategy to combine several different types of indicators.

In the field of optimization, an algorithm tries to find the best solution in a search space in terms of an objective function which should be minimized or maximized [8] accordingly. However, in singe-objective problems [9], there is only one objective to be optimized; in the multi-objective version, the algorithm tries to find a set of solutions based on more than one objective [10]. In the multi-objective optimization [11,12], the non-dominated sorting [13,14] is defined and used as a measure of efficiency in metaheuristic-based methods [15,16]. In [17], the basic dominance ranking was used to identify the excellent scientists according to all selected criteria. They selected all researchers in the first Pareto-front as excellent scientists, but by increasing the number of criteria (more than three) most compared entities were placed in the first Pareto front [17]. In this paper, we propose a modified, non-dominated sorting, which according to the basic dominance ranking, utilizes two main metrics and then two statistical metrics which are the computed means and medians of some ranks obtained by sorting each criterion's value in all compared vectors. This ranking has many major advantages: (1) it can perform very well at ranking all compared vectors even with a large number of criteria; (2) each obtained Pareto front in the modified non-dominated sorting has a smaller number of vectors in compared to the basic non-dominated sorting approach; (3) it can consider the length time of academic research (called the research period) as an independent indicator, which makes it possible to compare junior and senior researchers; (4) it is independent and capable of accommodating new indicators; (5) there is no need to determine the optimal weights to combine indicators. The modified Pareto dominance ranking was used to rank two research datasets with many criteria, ranking universities (200 samples) and countries (231 samples); additionally, the basic dominance ranking was applied to rank two research datasets with a low number of the criteria, ranking computer science researchers based on h-index and period of publication (350 samples) and ranking of universities based on triple rankings resources (100 samples).

The remaining sections of this paper are organized as follows. Section 2 presents a background review which provides state-of-the-art scientific indicators and ranking methods. Section 3 describes the proposed ranking method in detail. Section 4 presents case studies and corresponding discussions. Finally, the paper is concluded in Section 5.

2. Background Review

In this section, we review several state-of-the-art scientific indicators and several recent ranking methods.

2.1. A Brief Description of State-of-the-Art Scientific Indicators

Several indicators have been proposed to measure the scientific achievements. The pioneer studies introduced some basic indicators and described how these indicators can be combined to find the general intuition of the scientific outputs for researchers [18,19]. These indicators can be categorized in the following three main groups [20,21]:

- Production based indicators: these indicators were developed to assess the quantity of production such as the total number of published papers and the number of papers published during a limited time.
- Impact based indicators: they were proposed to quantify the impact of the researchers' publications; e.g., the total number of citations, the average number of citations per paper, the number of high-impact papers (papers with more than a specific number of citations), and the number of citations of the high-impact papers.
- Indicators based on the impact of the journals: these indicators were designed to consider journals where the papers are published; e.g., the median impact factor of the journals, relative citation rates (publication citations compared with the average citations of papers in the journal), and normalized position of the journals (computed according to position of journal in the ordered list in term of impact factor).

Some advantages and disadvantages of well-known indicators [6,22] are shown in Table 1.

Table 1. A summary of advantages and disadvantages for some commonly used indicators.

Indicator	Advantage	Disadvantage
The total number of published papers	It is a proper measure to quantify the productivity.	It does not consider the impact of their publications.
The total number of received citations	It can measure the total impact.	It may be inflated by a small number of "big hits" when a paper has many co-authors. It gives a Excess weight to highly cited survey papers.
Average number of citations per publication, without counting self-citations	It can be applied to compare junior and senior scientists (not in a complete way, because the senior researchers had more time for better building up of this metric).	It is hard to find and rewards low productivity and penalizes high productivity.
Number of "significant papers" (as the number of papers with having more than y citations)	It eliminates disadvantages of the previous mentioned indicators; the total number of published papers, the total number of citations, and average number of citations per publication.	The value of "y" should be adjusted.
The number of citations to each of the q most cited papers	Similar to Number of "significant papers," it can overcomes many of the mentioned disadvantages above.	"q" has not a single value so it is difficult to compute and compare.

In 2005, Hirsch dramatically changed scientometrics (informetrics) by introducing the h-index measure. Several studies have discussed and extended the validity of the h-index [23] since its introduction. The h-index has some significant properties [24,25]. It considers two aspects, the number of publications and their impacts on research. It performs better than other basic indicators (total number of papers, total number of citations, average number of significant papers, etc.) at evaluating

scientific achievements. In [25], an empirical study was conducted to confirm the superiority of the h-index over other basic indicators. In addition, the h-index can effortlessly be computed by using available resources such as the ISI Web of Science. Although it was extensively utilized as a scientometrics measure, it still suffers from the following drawbacks [1,26–28]:

- The h-index highly depends on the length of the academic career (the research period) because it is supposed the publications and citations of researchers increase over time. The h-index of new researchers has a very low value, and so it is not applicable for comparing scientists at different stages of their academic careers.
- It is field-dependent; therefore it can be useful to compare scientists in the same field of study.
- The h-index never decreases and also it may increase even if no new papers are published because the number of received citations for scientists can be increased with time. However, the value of h-index indicates the impact of the publications; it is strongly dependent on one aspect of the research; i.e., the age of research. In order to compare two scientists fairly based on their research achievements, in addition to quality evaluation, the period of time that they have researched over is also important. In other words, for two researchers with the same value of h-index, the researcher with shorter research period is the more successful researcher. Consequently, the h-index cannot be a standalone metric to assess the rank of a scientist in terms of different criteria.
- It is insensitive to performance changes because when first h articles received at least h times h, i.e., h^2 citations, it does not consider the number of citations they receive.
- Additionally, the h-index suffers from the same issues as other indicators, such as self-citations and being field-dependent. Some of these issues include difficulty in finding reference standards, and also problems of collecting all required data to compute the h-index (for example, discriminating between scientists with the same names and initials is challenging).

Several variants of the h-index have been developed to overcome the drawbacks of the h-index. The m-quotient [6] was proposed to account for years since the first publication, and it is computed as follows.

$$m\text{-quotient} = \frac{h\text{-index}}{n}, \tag{1}$$

where n is the number of years since the first published paper of the scientist. Batista et al. [29] introduced a complementary index as the h_I index which is defined by:

$$h_I = h^2/N_a^T, \tag{2}$$

where N_a^T is the number of authors in the considered h papers. In [30], A-index was suggested as the average number of citations of publications included in the h(Hirsch)-core which is mathematically defined as.

$$A = \frac{1}{h}\sum_{j=1}^{h} cit_j \tag{3}$$

The AR index [31] was proposed as the square root of the sum of the average number of citations per year of articles included in the h(Hirsch)-core. The mathematical definition of the index is as bellow.

$$AR = \sqrt{\sum_{j=1}^{h} \frac{cit_j}{a_j}}, \tag{4}$$

where a_j is the age of jth paper. Liang et al. [26] suggested a new index, the R-index, which found by calculating the square root of the sum of citations in the Hirsch core without dividing by h. This indicator is mathematically defined as.

$$R = \sqrt{\sum_{j=1}^{h} cit_j} \tag{5}$$

Egghe [28] introduced the g index which is defined as the highest number g of papers such that the top g papers together have at least g^2 citations. Additionally, it has proven that there is a unique g for any set of papers and $g > h$. Egghe and Rousseau [32] proposed the citation-weighted h-index (h_w-index) as follows.

$$h_w = \sqrt{\sum_{j=1}^{r_0} cit_j}, \; r_w(i) = \frac{\sum_{j=1}^{i} cit_j}{h}, \tag{6}$$

where cit_j is the number of the j-th most cited paper; r_0 is the largest row index i such that $r_w(i) \leq cit_i$. In general, even enhanced version of h-index metrics suffer from combining several metrics instead of considering them simultaneously.

2.2. A Brief Review of Ranking Methods

At the researcher level, all mentioned indicators can be applied to measure researchers' achievements. Although other scientific applications such as ranking scientific journals, research teams, research institutions, and countries tend to include a more comprehensive set of indicators, it is possible to apply the scientific indicators of researcher in other scientific comparative applications. For example, h-index can be calculated for an institute: "The h-index of an institute would be h_2 if h_2 number of its researchers have an h_1-index of at least h_2 each, and the other $(N - h_2)$ researchers have h_1-indices lower than h_2 each" [7]. In following, we briefly review some common ranking methods and indicators for universities. University rankings mainly use two different general categorizes of methodologies [33–39]; the first category uses all indicators [40,41] to calculate a single score, while the second category focuses more on a single dimension of university performance, such as the quality of research output [4], career outcomes of graduates [37], or the mean h-index [42]. The other indicators for university rankings are publication and citation counts, student/faculty ratio, percentage of international students, Nobel and other prize commonality, number of highly cited researchers and papers, articles published in Science and Nature, the h-index, and web visibility. First, some ranking methodologies of the first category are briefly described as below.

Liu and Cheng [43] proposed a ranking strategy, called Academic Ranking of World Universities (ARWU), which considers four measures: quality of education, quality of faculty, research output, and per capita performance. For comparison of four measures, the following six indicators are considered: (1) alumni of a university winning a Nobel Prize or a Fields Medal, (2) staff of a university winning a Nobel Prize or a Fields Medal, (3) highly cited researchers in 21 broad scientific fields, (4) publications in Nature and Science, (5) publications indexed in Web of Science, and (6) per capita academic performance of a university. It gives a score of 100 for the best performing university in each category and this university is considered as the benchmark against for computing the scores of all other universities. Then, the total scores of Universities are calculated as weighted averages of their individual category scores [44]. THE-QS World University Ranking (THE-QS) (http://www.topuniversities.com) was published by the Quacquarelli Symonds Company and considers six distinct indicators: academic reputation according to a large survey (40%), employer reputation (10%), the student faculty ratio (20%), citations per faculty based on the Scopus database (20%), the proportions of international professors (5%), and international students (5%). The World University Ranking was developed by Times Higher Education (www.timeshighereducation.co.uk/world-university-rankings) [41] which considers 13 indicators to rank universities. These indicators are categorized into five areas: teaching (30%), research (30%), citations (30%), industry income (2.5%), and international outlook (7.5%). They normalize the citation impact indicator to be suitable for different scientific output data.

Another global ranking is the Scimago Institutions Rankings (SIR) developed by the Scimago research group in Spain (www.scimagoir.com) [45]. SIR combines a quantity and various quality metrics. Indicators are divided into three groups: research output (total number of the

publication based on the Scopus database), international collaboration, leader output, high quality publications, excellence, scientific leadership (excellence with leadership, and scientific talent pool), innovation (innovative knowledge and technological impact), and societal (web size and the number of incoming links). The Cybermetrics Lab developed the Ranking Web of World Universities or Webometrics Ranking [46,47] which uses web data extracted from commercial search engines, including the number of webpages, documents in rich formats (pdf, doc, ppt, and ps), papers indexed by Google Scholar (indicator added in 2006), and the number of external in links as a measure of link visibility or impact. Higher Education Evaluation and Accreditation Council of Taiwan [48]) conducts university ranking which applies multiple indicators in the three categories: research productivity (the number of articles published in the past 11 years (10%) and the number of articles published in the current year (15%)), research impact (number of citations in the past 11 years (15%), number of citations in the past 2 years (10%), and average number of citations in the past 11 years (10%)), and research excellence (the *h*-index of the last 2 years (10%), the number of highly cited papers in the past 11 years (15%), and the number of articles of the current year in high impact journals (15%)). These rankings combine multiple weighted indicators to gain a single aggregate score to rank all universities. Additionally, some universities rankings [49,50] employed I-distance method [51] to apply all indicators for computing a single score as the rank. Besides its ability to calculate a single index (by considering several indicators) and consequently ranking countries, CIDI startegy utilizes the Pearson's coefficients of correlation, calculated using the I-distance method. In this case, the relevance of each input measure will be preserved. The I-distance method specifies the most important indicator instead of calculating numerical weights. The rank of indicator is determined by ordering them based on these correlations. In following, we mention some of ranking methodologies of the second category. The Centre for Science and Technology Studies at Leiden University published the LEIDEN Ranking (http://www.cwts.nl/ranking/LeidenRankingWebsite) [4,52] which has two main categories of indicators: impact and collaboration. The impact group includes three indicators: mean citation score, mean normalized citation score, and proportion of top 10% publications. The collaboration group includes four indicators: proportion of inter-institutional collaborative publications, proportion of international collaborative publications, proportion of collaborative publications with industry, and mean geographical collaboration distance. The Leiden Ranking considers the scientific performance instead of combining multiple indicators of university performance in a single aggregate indicator. U-Multirank [53,54] employs the variety of institutional missions and profiles and includes teaching and learning-related indicators. Additionally, it considers the importance of a user-driven approach in which the stakeholders/users are asked to determine indicators and their quality for ranking. In [37], they proposed a ranking methodology which considers only career outcomes of university graduates. This ranking focuses on the impact of universities on industry by their graduates. The mean h-index was used in [42] as a ranking metric to rank the chemical engineering, chemistry, materials science, and physics departments in Greece.

3. Proposed Methodology

As mentioned in the Section 2, several indicators and ranking methods have been proposed to measure the scientific achievements. There are two main categories of ranking methods: in the first one, the methods use all indicators (multi-metric) and in the second one, the methods focus on only one indicator (single-metric). Ranking methods by focusing on one indicator of scientific achievements cannot reveal significant differences among compared entities. In ranking methods with several indicators, first they need to assign weights for indicators which have considerable impacts on the results of these raking methods [55,56]. Finding the proper weights according to importance of indicators is a challenging task [57]. They also suffer from combining several different kinds of indicators to achieve a single score. In this paper, we modify the dominance depth ranking proposed in [13,14] utilized in the multi-objective optimization to rank scientific achievements. In 1964, Pareto [58] proposed the Pareto optimality concept, which has been applied in a wide range of

application, such as economics, game theory, multi-objective optimization, and the social sciences [59]. Pareto optimality was mathematically defined as a measure of efficiency in the multi-objective optimization [12,60]. We explain Pareto optimality concepts and also the proposed method and how it can be applied to evaluate scientific achievements. Without loss of generality, it is assumed that the optimal value of each criterion as a preference be a minimal value. Seeking the optimal value among both the minimal and maximal values is analogous, and if a criterion value element C_i to be maximized, it is equivalent to minimize $-C_i$.

In the following, the Pareto optimality definitions are described by the assumption of the minimal value as the optimal.

Definition 1 ((Pareto Dominance) [61]). *A criterion vector $u = (u_1, \ldots, u_n)$ dominates another criterion vector $v = (v_1, \ldots, v_n)$ (denoted as $u \prec v$) if and only if $\forall i \in \{1, \ldots, n\}, u_i \leq v_i$ and $u \neq v$. This type of dominance is called weak dominance in which two vectors can be same in some objectives, but they should be different in at least one objective. However, in strict dominance, u has to be better on all objectives; i.e., it can not have the same objective value with v.*

The Pareto optimality concept is defined from the dominance concept as follows.

Definition 2 (Definition (Pareto Optimality) [61]). *A criterion vector u in a set of criterion vectors (S) is a Pareto optimal vector (non-dominated) if for every vector x, x does not dominate u, $x \preceq u$.*

Figure 1 shows Pareto optimal solutions and dominated solutions for a criterion value vectors (2D) (f_1, f_2). According to this definition, for a set of objective function vectors or criterion value vectors, the Pareto set is denoted as all Pareto optimal vectors which have no elements (criterion values) that can be decreased without simultaneously causing an increase in at least one of the other elements of vectors (assuming a Min-Min case).

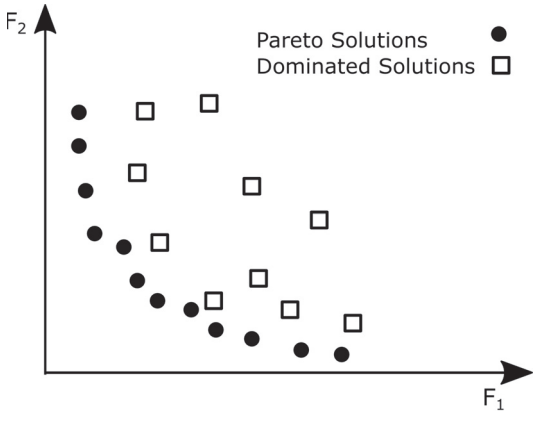


Figure 1. Pareto optimal set (non-dominated solutions) and dominated solutions for a two dimensional space.

Definition 3 (Definition (Pareto-front) [61]). *For a given set S, the Pareto front is defined as set $S\ \{x \in S | \nexists y \in S, y \prec x\}$.*

Figure 2 shows the Pareto front for two dimensional space for all four possible cases for minimizing or maximizing of two objective function vectors (f_1, f_2) or a two criterion value vectors (f_1, f_2).

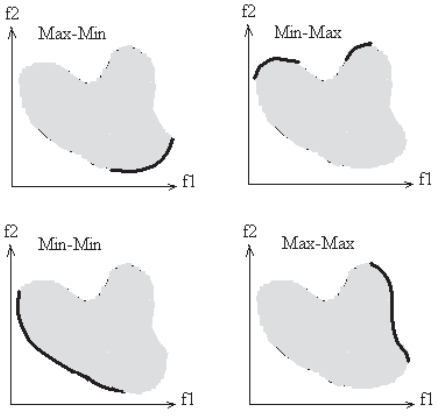


Figure 2. Pareto front for a two dimensional space.

Dominance depth ranking in the non-dominated sorting genetic algorithm (NSGA-II) was proposed by Deb et al. [13] to partition a set of objective function vectors (criterion value vectors) into several clusters by Pareto dominance concept. First, the non-dominated vectors in a set of criterion value vectors assigned to rank 1 and form the first Pareto front (PF1), and all these non-dominated vectors are removed. Then, non-dominated solutions are determined in the set and form the second Pareto front (PF2). This process is repeated for other remaining criterion value vectors until there is no vector left. Figure 3 illustrates an example of this ranking for a set of eight points (criterion value vectors) and Table 2 shows the coordinates of points. First points 1, 2, 3, and 4 as non-dominated solutions are ranked to rank 1. Then, for the rest of the points (points 5, 6, 7, and 8), non-dominated solutions are determined so points 5 and 6 as non-dominated solutions are ranked as 2 and removed. In the last iteration, the remaining points 7 and 8 are ranked as rank 3. The details of non-dominated sorting algorithm is presented in Algorithm 1.

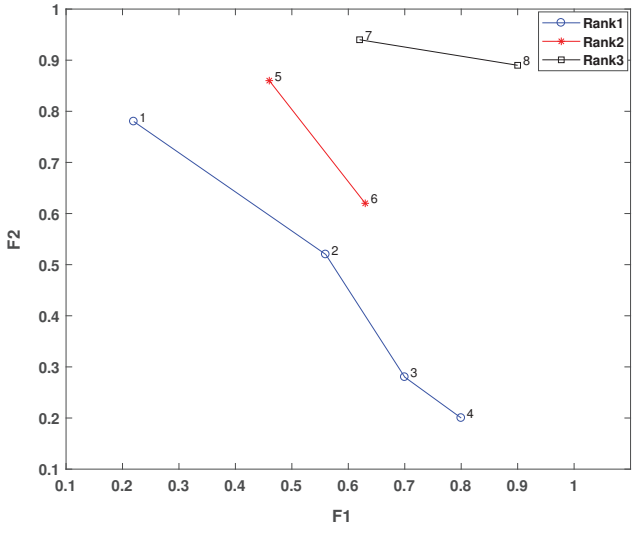


Figure 3. An example of a dominance depth ranking method.

Algorithm 1 Non-dominated sorting algorithm.

Input : V: Set of criteria vectors, N: The number of vectors
Output: Perato fronts ranks

while $N \neq 0$ **do**
  for $i \leftarrow 1$ **to** N **do**
    $n_i = 0$ **for** $j \leftarrow 1$ **to** N **do**
      `// Calculating the number of vectors that dominate` v_i
      if $v(j) \prec v(i)$ **then**
        $n_i = n_i + 1$
      end
    end
    if $n_i = \emptyset$ **then**
      $F_i = F_i \bigcup v(i)$
    end
  end
  `// Temporarily removing Pareto front from set to compute the next fronts`
  $V = V - F_i \;\; N = N - size(F_i)$
end

Table 2. A numerical example of computed new metrics for eight points shown in Figure 3. Four new statistical metrics are mean-ranks, median-ranks, dominated number, and nn-dominated number. Ranks-F1 and Ranks-F2 are ranks (two columns Ranks-F1 and Ranks-F2) for two criterion vectors F1 and F2.

Point	F1	F2	Ranks-F1	Ranks-F2	Mean-Ranks	Median-Ranks	Non-Dominated Number	Dominated Number
1	0.22	0.78	1	5	3	3	0	3
2	0.56	0.52	3	3	3	3	0	3
3	0.7	0.28	6	2	4	4	0	1
4	0.8	0.2	7	1	4	4	0	1
5	0.46	0.86	2	6	4	4	1	2
6	0.63	0.62	5	4	4.5	4.5	1	1
8	0.9	0.89	8	7	7.5	7.5	3	0
7	0.62	0.94	4	8	6	6	6	0

In [17], the dominance concept was used to identify the excellent scientists whose performances cannot be surpassed by others with respect to all criteria. The proposed method can provide a short-list of the distinguished researchers in the case of award nomination. It computes the sum of all criteria and sorts all researchers according to this calculated sum value. After that, the researcher with the maximum sum r_{max} is placed in the skyline set. The second best researcher is compared with the researcher in the skyline set ($r_{skyline}$); if he/she is not dominated by r_{max}, he/she is added into the skyline set. This process is repeated for all remaining researchers to construct the skyline set: if they are not dominated by all researchers in the skyline set ($r_{skyline}$), then they are added into the skyline set. In fact, they select all researchers in the first Pareto front using the dominance concept. There is a well-known problem with the first Pareto created by the basic non-dominated sorting [17]. By increasing the number of criteria (more than three criteria) in the set of the criterion value vectors, a large number of the compared vectors become non-dominated vectors and are placed in the first Pareto front. By increasing the number of criteria, the chance of placing a criterion value vector while having only one better criterion value in the first Pareto front is increased. In order to demonstrate this problem, Table 3 shows three Pareto fronts by the non-dominated sorting for countries data extracted from the site "http://www.scimagojr.com" including five indicators: citable documents (CI-DO), citations, self-citations (SC), citations per document (CPD), and h-index; Table 3 shows the results of the non-dominated sorting method. As it can be seen from Table 3, three countries, Panama, Gambia,

and Bermuda, are in the first Pareto front because they have higher values for only one criterion indicator (CPD) while other criteria values are low. Additionally, Montserrat has the rank 2 because it has the high value for only the CPD indicator.

Table 3. Indicators and the Pareto fronts from one to three by the non-dominated sorting on the country data.

Country	Rank	Documents	CI-DO	Citations	CPD	*h*-Index
United States	1	9,360,233	8,456,050	202,750,565	21.66	1783
Netherlands	1	746,289	682,627	16,594,528	22.24	752
Switzerland	1	541,846	501,917	12,592,003	23.24	744
Panama	1	5129	4830	137,585	26.82	142
Gambia	1	2004	1859	54,925	27.41	99
Bermuda	1	633	590	21,884	34.57	73
China	2	4,076,414	4,017,123	24,175,067	5.93	563
United Kingdom	2	2,624,530	2,272,675	50,790,508	19.35	1099
Sweden	2	503,889	471,036	10,832,336	21.5	666
Denmark	2	290,994	269,364	6,405,076	22.01	558
Iceland	2	15,625	14,353	357,678	22.89	218
Montserrat	2	95	93	2282	24.02	27
Germany	3	2,365,108	2,207,765	40,951,616	17.31	961
Canada	3	1,339,471	1,227,622	25,677,205	19.17	862
Israel	3	295,747	274,748	5,826,878	19.7	536
Faroe Islands	3	510	472	10,105	19.81	48
Guinea-Bissau	3	458	421	9357	20.43	50

In this paper, we propose a modified non-dominated sorting (described in Algorithm 2) to rank the scientific data. First we use the dominance depth ranking for all vectors; after that for each criterion value vector two new statistical metrics are calculated. For each vector, two metrics are the dominated number and the non-dominated number which show the number of the dominated vectors by this vector and the number of vectors which dominate this vector. Additionally, we used two other statistical measures proposed in [62]. These statistical measures are computed to sort the criterion value vectors. In [62], first for each criterion value C_i, all vectors are sorted according to this criterion value C_i in ascending order and their ranks are assigned based on their sorting order. After that, for each criterion value vector some statistical measures like the minimum of its rank or the sum of its rank are used to make Pareto fronts.

We also sort all vectors according to each criterion value and calculate the ranks of vectors corresponding this sorting; after that we compute the mean and median of ranks of each vector as two new metrics. Table 2 shows an example of computed new metrics for eight points in Figure 3. F_1 and F_2 are the values of sample points in Figure 3 which are considered just as the numerical examples for a two-objective problem. For each point, ranks (two columns Ranks-F1 and Ranks-F2) for two criterion vectors (F_1, F_2) are computed according to their sorting order. Thus, we have four new statistical metrics (the mean and median of ranks, also the dominated number and the non-dominated number) which we use as criteria (objectives) to measure various levels of scientific achievement by applying dominance depth ranking again to make all Pareto fronts. We used the basic non-dominated sorting for data with two and three criteria and the modified non-dominated sorting for the data with more than three criteria. The proposed method has major advantages that are described in detail. In this method, vectors with one better criterion value than others cannot move toward the first front.

Additionally, increasing the number of criteria cannot negatively influence the obtained ranks (no big portion of entities in the first front, as before); each rank corresponding to a Pareto front has a smaller number of vectors, so in total it assigns more ranks to the criterion vectors.

Algorithm 2 Modified non-dominated sorting algorithm

Input : V: Dataset including criteria vectors.
Output: *Ranks*: Ranks of all criteria vectors

```
// determine the dominance relation on each pair of criterion vector.
```
Nd_i=Number of vectors which dominate ith vector, v_i in V;
NNd_i=Number of dominated vectors by ith vector, v_i in V;

```
// compute two new metrics
```
for $c \leftarrow 1$ **to** M **do** `// For each criterion`
  `// sort all values of each criteria over all vectors in ascending order.`
  $SortedF_c$=Sort(F_c);
  for $i \leftarrow 1$ **to** N **do** `// For each vector in` V
    `// compute the rank of` i`th vector based on` c`th criterion.`
    $Rank(i, c)$=Index of v_i in $SortedF_c$;
  end
end
for $i \leftarrow 1$ **to** N **do** `// For each vector in` V
  $Meanranks_i$=Mean($SortedF(i, :)$);
  $Medianranks_i$=Median($SortedF(i, :)$);
end
`// generate ranks of vectors based on the old and new metrics by Algorithm. 1`
$Ranks \leftarrow$ Non-dominated sorting algorithm($F_1, ..., F_M, Nd, NNd, Meanranks, Medianranks$);

In order to demonstrate the performance of this modified non-dominated sorting, Table 4 shows four Pareto fronts by the modified non-dominated sorting for extracted country data. Because the considered criteria have different scales, in all experiments, in order to apply the proposed method, they are normalized. As can be seen in the first Pareto front, only the United States is placed and Panama is in the forth Pareto front. Additionally, other countries with only one high criterion value, Gambia and Bermuda, which are in the first Pareto front by the non-dominated sorting method (as it can be seen in Table 3) are not placed in four Pareto fronts obtained by the modified non-dominated sorting method. Additionally, it can be seen that the number of countries in each Pareto front by using the modified non-dominated sorting is smaller than in basic non-dominated sorting.

In addition, we consider the period research as a new criterion value. Using Pareto dominance ranking makes it possible to have the research period as an independent indicator to be considered for ranking the scientific data. Considering the research period as the indicator provides a predication mean for some research cases. For example, suppose for comparing authors, criterion values be h-index and the research period $A_i = (h - index, time)$: two authors $A_1 = (80, 40)$ and $A_2 = (20, 10)$ would be in the same Pareto front because based on Pareto optimality concept, they do not dominate each other; therefore, we can predict that the author A_2 probably will be able to have the same performance as the author A_1 (or even better) after some years. According to observed values of indicators for universities, authors, and countries, this method can be utilized for prediction of their future performance. Additionally, the time length indicator enhances this ranking method with a traceable feature; that means by collecting data during times, we can observe how the performances of universities or researchers change and if they can improve their Pareto front ranks or not. In addition, this method can be applied to compute ranks by using obtained ranks from other ranking methods (ranking by multiple resources). In this way, each indicator is an obtained rank from a ranking method and it is expected that the non-dominated vectors in the first Pareto front contain the vectors with the minimum/maximum values of indicators, for Min-Min or Max-Max cases, respectively. Pareto dominance ranking can take into account any new kind of indicator as a new criterion value.

Table 4. The Pareto fronts from one to forth by the proposed method on the country data.

Country	Rank	Documents	CI-DO	Citations	CPD	*h*-Index
United States	1	9,360,233	8,456,050	202,750,565	21.66	1783
Netherlands	2	746,289	682,627	16,594,528	22.24	752
United Kingdom	2	2,624,530	2,272,675	50,790,508	19.35	1099
Switzerland	3	541,846	501,917	12,592,003	23.24	7444
China	3	4,076,414	4,017,123	24,175,067	5.93	563
Germany	3	2,365,108	2,207,765	40,951,616	17.31	961
Canada	3	1,339,471	1,227,622	25,677,205	19.17	862
Panama	4	5129	4830	137,585	26.82	142
Sweden	4	503,889	471,036	10,832,336	21.5	666
Denmark	4	290,994	269,364	6,405,076	22.01	558
Iceland	4	15,625	14,353	357,678	22.89	218
Japan	4	2,212,636	2,133,326	30,436,114	13.76	797
France	4	1,684,479	1,582,197	28,329,815	16.82	878

4. Experimental Case Studies and Discussion

We run the basic Pareto dominance ranking on the following scientific data with two and three criteria and modified Pareto dominance ranking on the following scientific data with more than three criteria. The first dataset includes 350 top computer science researchers (http://web.cs.ucla.edu/~palsberg/h-number.html) which contains a partial list of computer science researchers who each has an *h*-index of 40 or higher according to the Google Scholar report. This data has two indicators: research period (a low value is better) and *h*-index (a high value is better). The *h*-index values were collected from Google Scholar for the year 2016 and research period values were calculated from the year of the first publication of an author so far. The second dataset includes the 200 top universities ranked by URAP (a nonprofit organization (http://www.urapcenter.org)). This dataset has six indicators: article, citation, total document (TD), article impact total (AIT), citation impact total (CIT), and international collaboration (IC). The third dataset has 231 top countries (for the year 2015) extracted from the site SJR (http://www.scimagojr.com), including six indicators: documents, citable documents (CI-DO), citations, self-citations (SC), citations per document (CPD), and *h*-index. We do not consider the SC indicator because it is not certain that the maximum value or minimum value of this value is desirable. The forth dataset consists of the three ranks of 100 top common universities collected from three resources; the QS World University Rankings (https://www.topuniversities.com), URAP (http://www.urapcenter.org), and CWUR Rankings (http://cwur.org). In the following, we report all results of mentioned approaches on the four datasets in detail.

4.1. The First Case Study: Ranking Researchers

Table A1 indicates the names of researchers, research period, *h*-index, and the obtained Pareto ranks from the basic Pareto dominance ranking (Pareto ranking). From Table A1, it can be seen that first Pareto ranks include researchers with high values of *h*-index and low research period values. For instance, the researcher "Zhi-Hua Zhou" has the minimum value of research period 14 and the researcher "A. Herbert" has the maximum value of *h*-index, 162. Researchers in the first Pareto front are A. Herbert, K. Anil, Han Jiawei, Van Wil, Buyya Rajkumar, Perrig Adrian, and Zhou Zhi-Hua; the second Pareto front contains Shenker Scott, Foster Ian, Salzberg Steven, Schlkopf Bernhard, Schmid Cordelia, Abraham Ajith, and Xiao Yang. Additionally, it can be observed that researchers with the maximum value of research year indicator (40) are associated with the higher rank because they are dominated by other researchers according to Pareto dominance concept. Researchers having

values close to the value of h-index 52 or higher are associated with the higher rank due to the Pareto dominance concept. Figure 4 shows the ranks in terms of Pareto fronts for all researchers. It can be seen from Figure 4 that the extent of improvement for a researcher A_i can change his/her Pareto front ranking by looking at researchers which dominate A_i and are located in the better Pareto fronts.

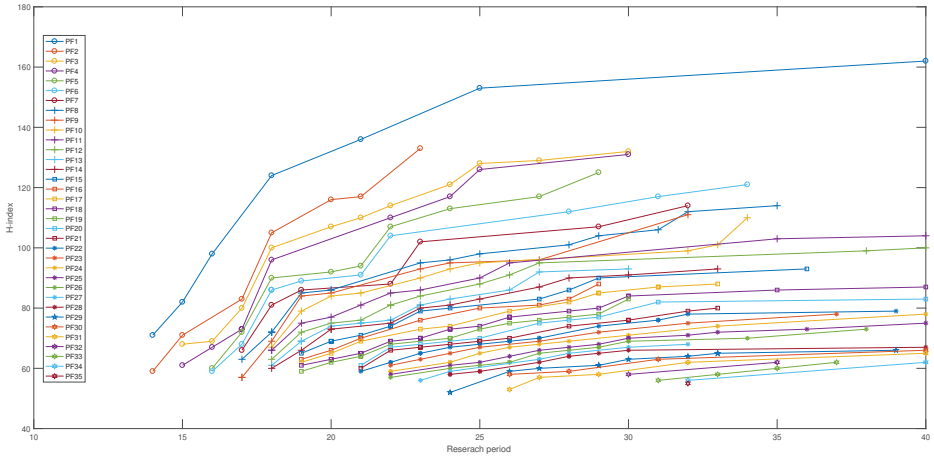
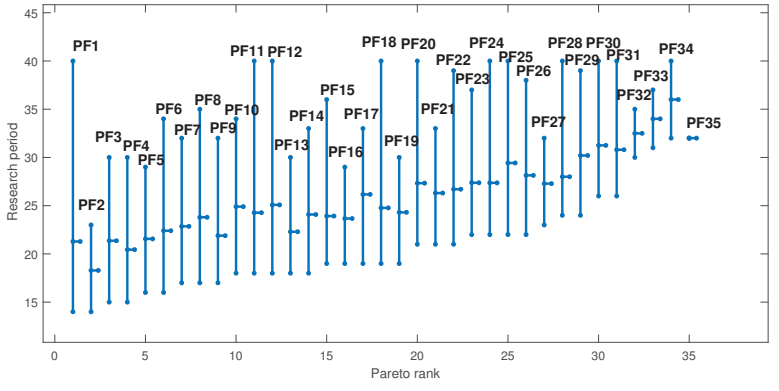
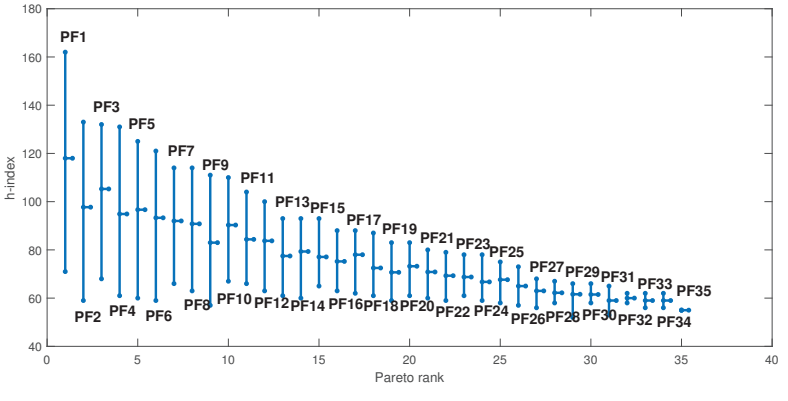


Figure 4. Pareto fronts for the researcher dataset. Different colors and symbols are used to distinguish thirty five Pareto fronts with two research period (the horizontal axis) and h-index (the vertical axis) indicators.

To gain a better understanding of the Pareto ranking with each indicator, we plot the obtained Pareto ranks from the first rank to the thirty fifth versus each indicator. In Figure 5, vertical lines demonstrate Pareto ranks from the first rank to the thirty fifth, which at the top of each line indicates the maximum value of the indicator; its bottom is the minimum value of the indicator; and the short horizontal tick in the middle of each line is the average value of the indicator. Figure 5 indicates that the research period of the first Pareto front includes values with the maximum and minimum of the length time. That is reasonable because it is expected that authors who have had more time have higher h-index values so they could be located in the first Pareto front, and younger authors having had shorter research periods and reasonable h-index values also could be in the first Pareto front. The average values of the research period for the beginning Pareto fronts are low values while the last Pareto fronts have higher average values. From Figure 5, we can see that the maximum, average, and minimum of h-index values for Pareto fronts decrease from the first Pareto front to the 35th. Additionally, the first Pareto front has the maximum h-index values and the last Pareto front includes the minimum h-index values.



(**a**)



(**b**)

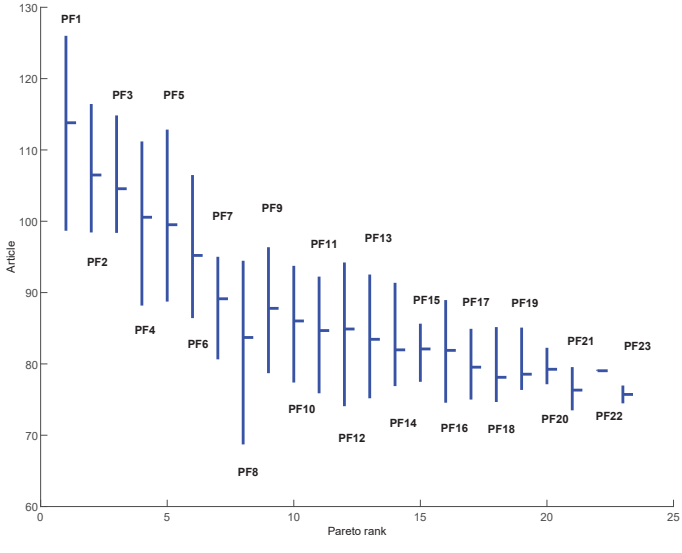
Figure 5. The rank of *h*-index and research period values for Pareto fronts in the researcher data. (**a**) Research period; (**b**) *h*-index.

4.2. The Second Case Study: Ranking of Universities

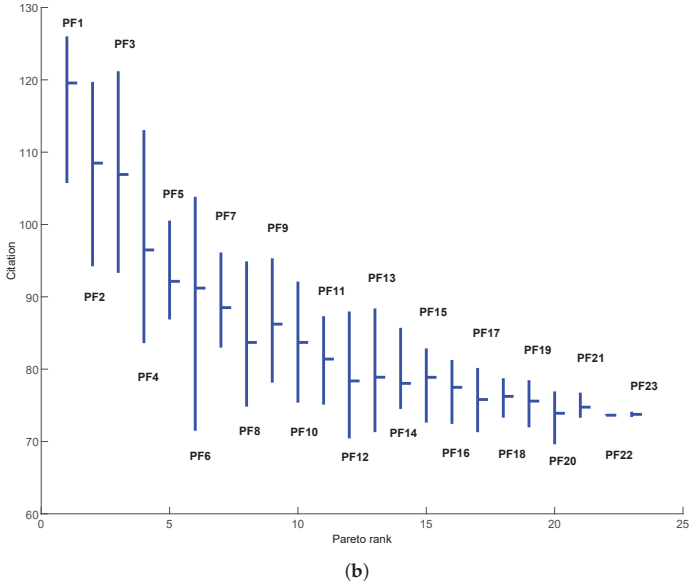
Six indicators of university dataset and their ranks obtained by modified Pareto dominance ranking are summarized in Table A2. As mentioned in Section 3, for fair comparison, we add the time period of academic research (the research period (RP)) mentioned in Table A2 as an indicator in the data which is calculated as the length of the university established year to present. Based on the proposed method, the first Pareto front has six universities, including top universities; for example, Harvard University, University of Toronto, and Stanford University. In the basic Pareto dominance ranking, the first Pareto front has twenty universities. Additionally, the proposed ranking clusters this data into twenty three Pareto fronts but the Pareto dominance ranking has only eight Pareto fronts. As was mentioned in the Section 3, the proposed method can assign more ranks to the criterion vectors even by increasing the number of criteria (many-metric cases).

In order to deep understand the behavior of the obtained Pareto ranks and indicators, we plot the maximum, minimum, and average of values for all indicators versus Pareto ranks in Figures 6–8 as mentioned before. It can be seen from these figures—all plots for six indicators—that there is a decreasing behavior in terms of the maximum, minimum, and average values, observable from the

first Pareto front to the last Pareto front. In addition, Figure 9 visualizes universities in the four top ranked Pareto fronts. Each line illustrates one university (a five dimensional vector) in which the values of five indicators are presented using vertical axes; i.e., coordinate's value.

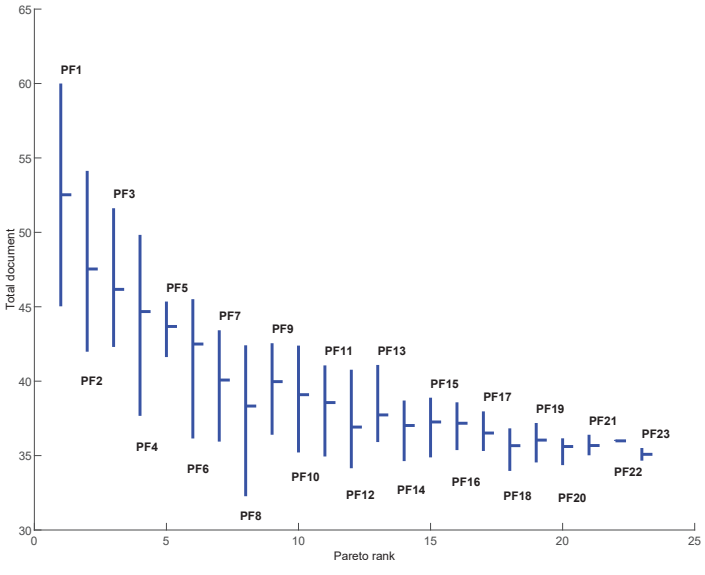


(**a**)

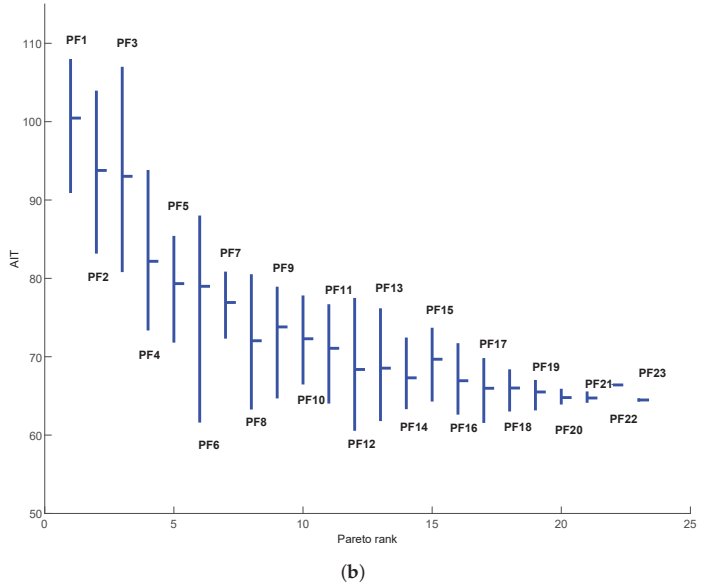


(**b**)

Figure 6. The rank of article and citation indicators for universities based on each Pareto front in the university data. (**a**) Article; (**b**) citation.

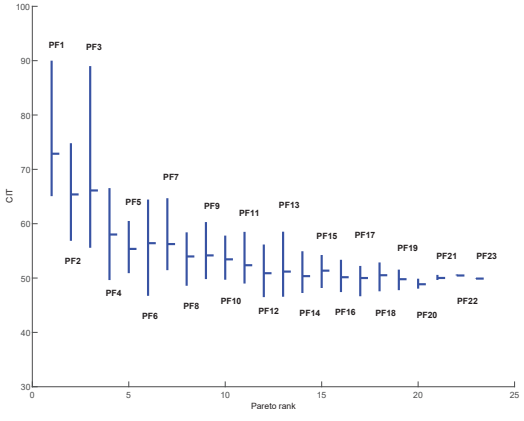


(**a**)

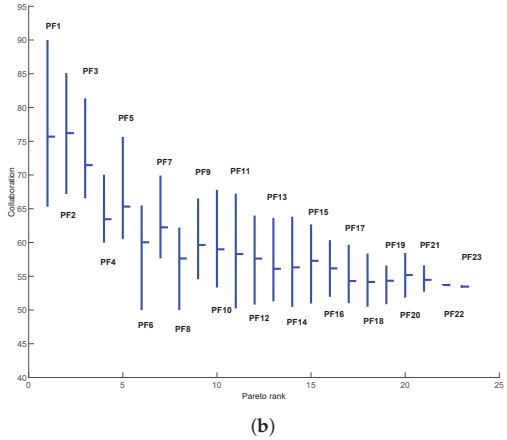


(**b**)

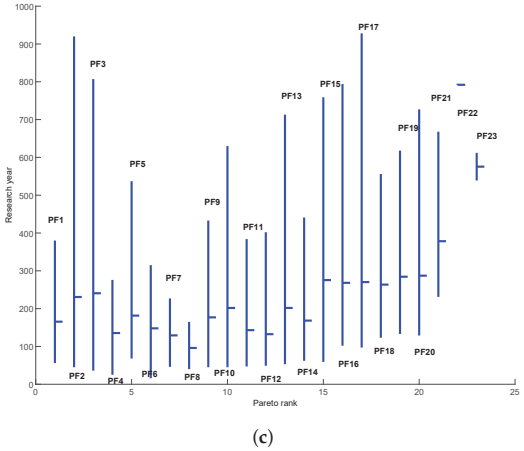
Figure 7. The rank of total document and article indicators for universities based on each Pareto front in the university data. (**a**) Total document; (**b**) article impact total (AIT).



(**a**)



(**b**)



(**c**)

Figure 8. The rankk of each indicator for universities based on each Pareto front for the university data. (**a**) Citation impact total (CIT); (**b**) collaboration; (**c**) research period.

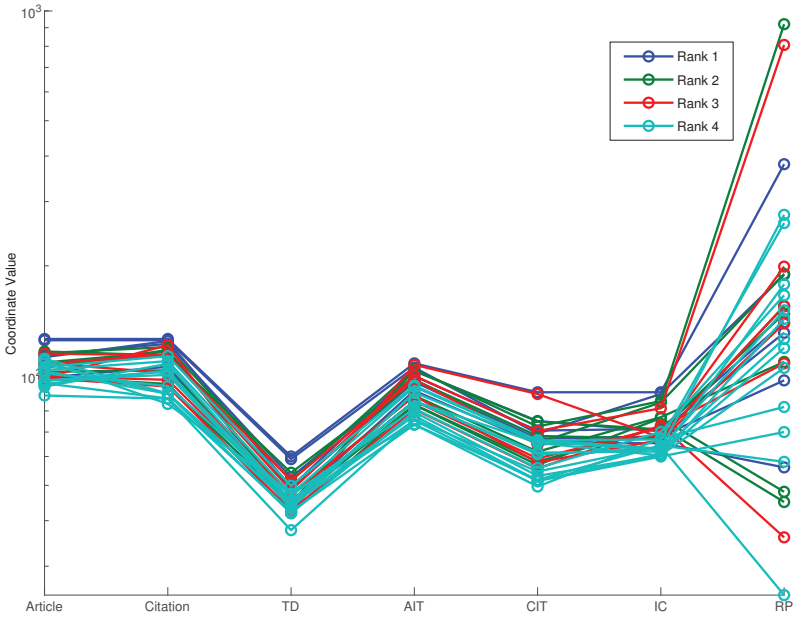
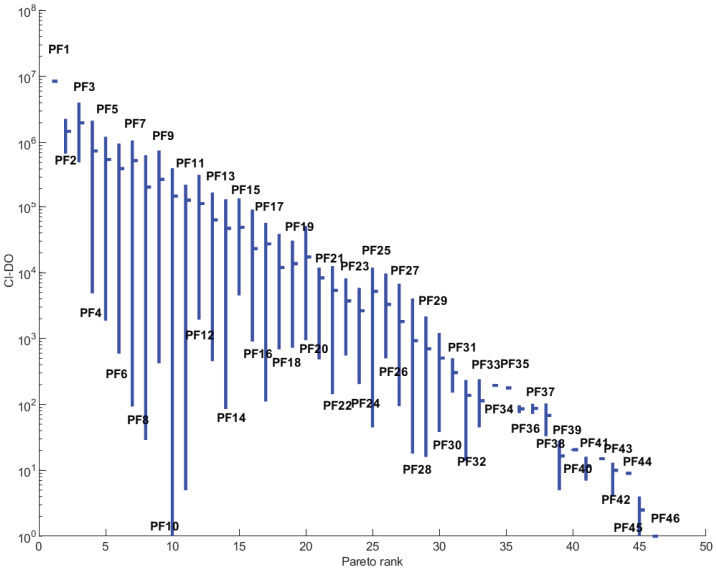


Figure 9. The Parallel coordinates for Pareto fronts one to four for the university data.

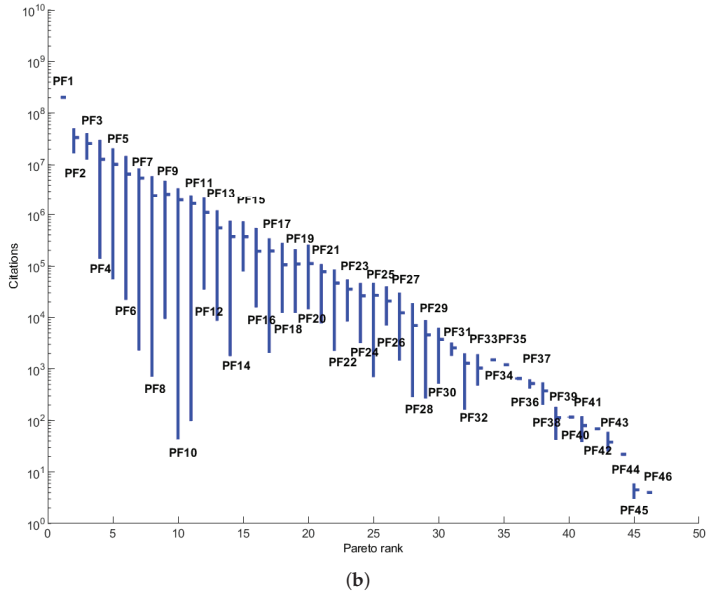
4.3. The Third Case Study: Ranking of Countries

Table A3 shows countries, the values of five indicators (documents, CI-DO, citations, CPD, and *h*-index), and the obtained Pareto ranks from the proposed method (Pareto ranking). The United States is located in the first Pareto front because it has the maximum values of four indicators: documents, CI-DO, citations, CPD, and *h*-index. The United States is assigned to the rank 1 and in the second Pareto front, Switzerland and the United Kingdom are placed. The proposed method ranks these countries into forty six ranks while in the Pareto dominance ranking, it has thirty Pareto fronts.

Additionally, for this data, we plot the maximum, minimum, and average of values for all indicators versus Pareto ranks in Figures 10 and 11. Figures show a falling tendency of the average values from the first Pareto front to the last Pareto front. Additionally, we compute the percentage of the number of countries from the different continents (Asia, Europe, Latin America, Middle East, North America, and Pacific region) for each Pareto front. Figure 12 shows the percentage number for each continent. In Figure 12, the first largest and second largest percentages of the first Pareto front are North America and Europe. In addition, Figure 13 visualizes the values of indicators for countries in the four top ranked Pareto fronts by the parallel coordinates visualization technique. Each line illustrates one country (a five dimensional vector) in which the values of five indicators are presented using vertical axes; i.e., coordinate's value. For instance, the value of CI-DO indicator is in interval $[1, 10^7]$ for countries on the four first Pareto fronts.

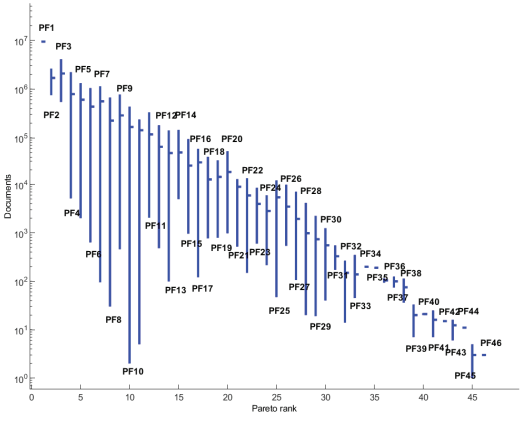


(**a**)

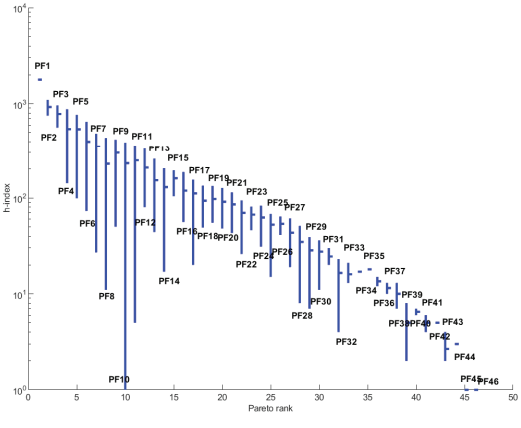


(**b**)

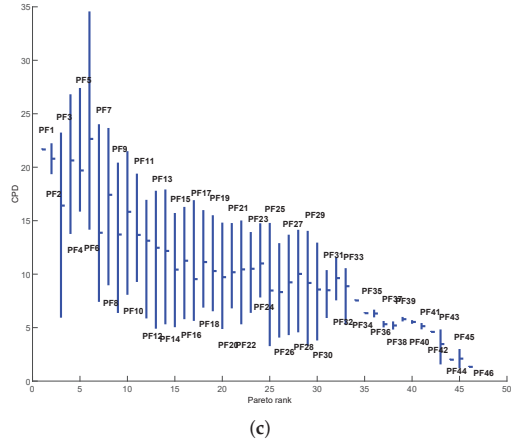
Figure 10. The rank of each indicator for countries based on each Pareto front for the country data. (**a**) Citable documents (CI-DO); (**b**) citations.



(**a**)



(**b**)



(**c**)

Figure 11. The rank of each indicator for countries based on each Pareto front for the country dataset. (**a**) Document; (**b**) *h*-index; (**c**) citations per document (CPD).

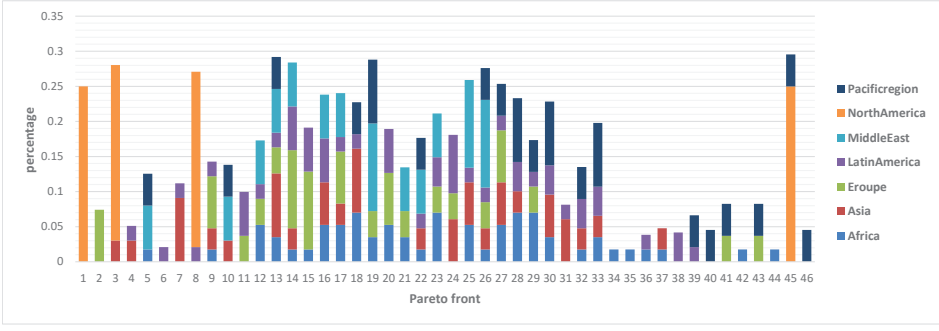


Figure 12. The number of countries in each continent for all Pareto fronts in the third case study.

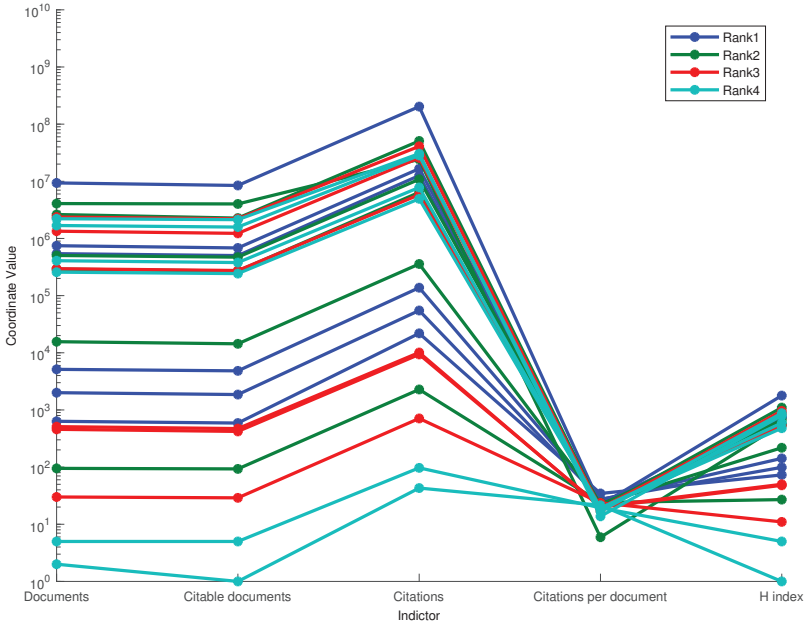


Figure 13. The parallel coordinates for Pareto fronts one to four for the country dataset.

4.4. The Forth Case Study: Resolution for Multi-Rankings of Universities

This case study collects the three ranks of 100 top common universities collected from the three mentioned resources, from which it is supposed that the criterion vectors with the lesser values for all three ranks are better vectors (i.e., Min-Min-Min). Table A4 shows universities, the values of three ranks, and the obtained Pareto ranks from Pareto dominance ranking (Pareto ranking). As we can see, three universities, "Massachusetts Institute of Technology," "Stanford University," and "Harvard University" are located in the first Pareto front, which has elements with the values 1 and 2 as the obtained ranks from other ranking resources. Figure 14 shows the numbers of Pareto fronts for all data. Additionally, the maximum, minimum, and average of values for three rankings versus Pareto ranks

are plotted in Figure 15. It can be seen from Figure 15 that the average values of three ranks increase from the first Pareto front to 13th Pareto front.

At the end of this section, several points regarding the performance of the method and its differences with other ranking strategies are mentioned. First of all, a multi-criteria indicator is proposed for ranking the researchers, universities, and countries. Considering two or more objectives simultaneously can provide a fairer ranking. For instance, using research period along with other important criteria provides a fair comparison for senior and junior researchers to discover more-talented researchers. Secondly, since the considered criteria to assess the entities are different from indicators in other ranking strategies, the resultant rankings are completely different. In fact, they evaluate the universities in terms of different metrics. As a result, the comparison between the results of ranking strategies does not lead to a precise and meaningful conclusion. On the other hand, the proposed method clusters the entities based on multiple criteria into different levels. Accordingly, all universities in the same Pareto are ranked equally; for instance, based on this perspective, all universities in the first Pareto are the top ranked universities. Finally, this method does not actually define an evaluation measure; it gives a strategy to rank not only the case studies in the paper, but also any multi-criteria data entities. In addition, using this general platform provides the chance to utilize any metric to assess the related entities without modification to other parts of the algorithm.

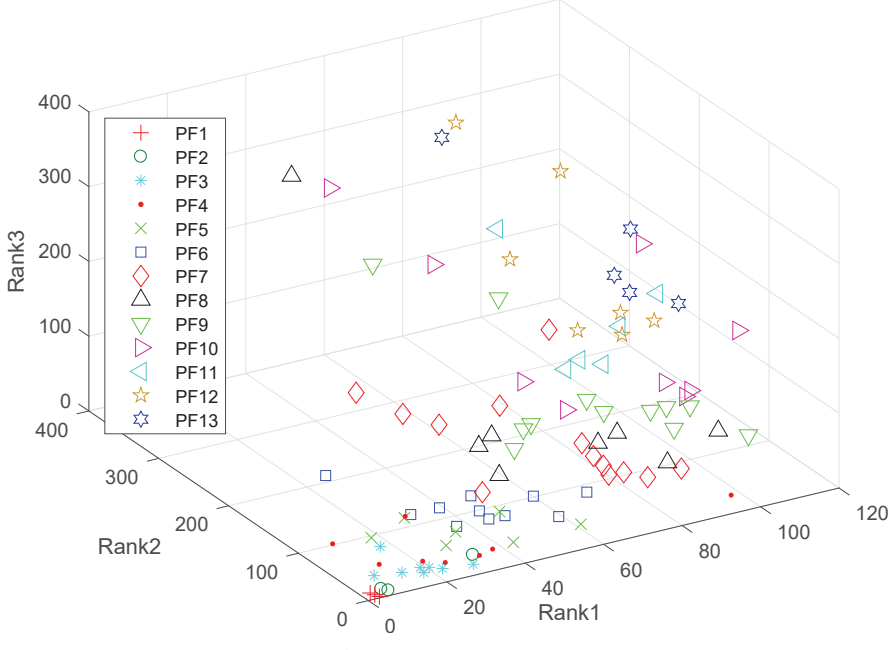
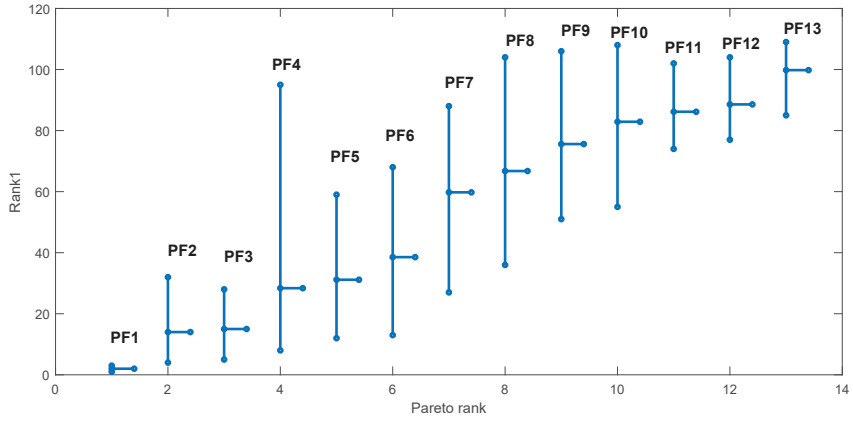
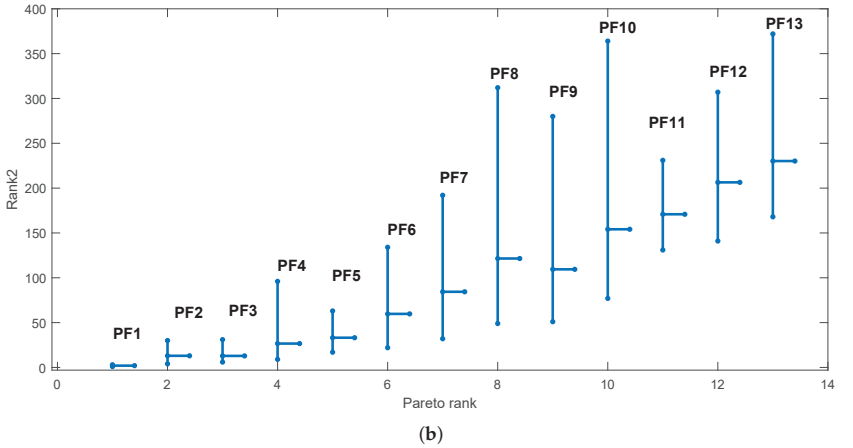


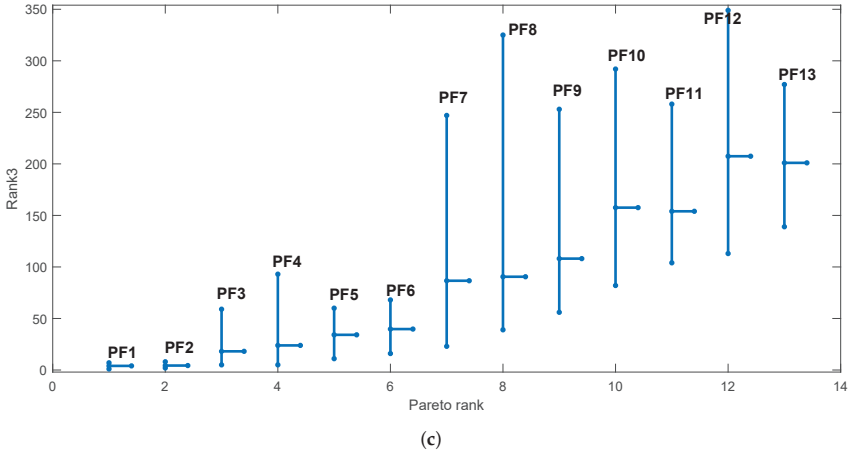
Figure 14. Pareto fronts obtained by using three ranks.



(**a**)



(**b**)



(**c**)

Figure 15. The rank of ranks based on each Pareto front for the ranks of universities data. (**a**) Rank1; (**b**) Rank2; (**c**) Rank3.

5. Conclusions and Future Directions

In this paper, a modified Pareto-front based ranking was suggested as a new ranking method for measuring the scientific achievements, or in general multi and many- metric rankings. By using some dominance metrics obtained from the basic Pareto dominance depth ranking and some statistical metrics sorting compared criteria, the proposed method is able to find some different groups (clubs) for entities of a dataset having a large number of the criteria. It provides simultaneously multiple comparisons, considering the time period of academic research, and the use of other ranking methods. We selected different kinds of the scientific datasets; namely, computer science researchers, top universities, countries, and multiple rankings of universities to rank by using Pareto ranking. In future, we are planning to develop ranking strategies based on other dominance-based rankings; for example, dominance rank [61,63] which is related to the number of data entries in the set which dominates the considered point. Finally, we are interested in considering the use of other types of domination definition, such as the concepts of weak dominance, strict dominance, and ϵ-dominance. Additionally, many (more than three) metrics and various resources will be studied in the future.

Author Contributions: Data curation, S.M.; Formal analysis, S.R. and S.M.; Methodology, S.R.; Project administration, S.R.; Software, S.M. and A.A.B.; Supervision, S.R. and K.D.; Validation, S.R. and K.D.; Visualization, S.M. and A.A.B.; Writing—original draft, S.M.; Writing—review & editing, S.R. and A.A.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Indicators and Pareto ranks for the author data. Indicators are *h*-index and the research period (RP).

Author	Rank	RP	H-Index	Author	Rank	RP	*h*-Index
A.Herbert	1	40	162	Burgard Wolfram	6	19	89
K.Anil	1	25	153	MllerKlaus Robert	6	18	86
Han Jiawei	1	21	136	Horrocks Ian	6	18	86
van Wil	1	18	124	Liu Bing	6	17	68
Buyya Rajkumar	1	16	98	Harman Mark	6	16	59
Perrig Adrian	1	15	82	Dongarra Jack	7	32	114
ZhouZhi-Hua	1	14	71	A.John	7	29	107
Shenker Scott	2	23	133	Nayar Shree	7	23	102
Foster Ian	2	21	117	SeidelHans-Peter	7	22	88
Salzberg Steven	2	20	116	Rexford Jennifer	7	19	86
Schölkopf Bernhard	2	18	105	Govindan Ramesh	7	18	81
Schmid Cordelia	2	17	83	Gao Wen	7	17	66
Abraham Ajith	2	15	71	Grossberg Stephen	8	35	114
Xiao Yang	2	14	59	Dubois Didier	8	32	112
Sejnowski Terrence	3	30	132	H.Randy	8	31	106
Haussler David	3	27	129	Horowitz Mark	8	29	104

Table A1. *Cont.*

Author	Rank	RP	H-Index	Author	Rank	RP	*h*-Index
I.Michael	3	25	128	Osher Stanley	8	28	101
Zisserman Andrew	3	24	121	Szeliski Richard	8	25	98
Estrin Deborah	3	22	114	H.Vincent	8	24	96
Koller Daphne	3	21	110	Malik Jitendra	8	23	95
Herrera Francisco	3	20	107	B.Mani	8	20	86
Balakrishnan Hari	3	18	100	Baraniuk Richard	8	20	86
Staab Steffen	3	17	80	Fox Dieter	8	19	85
Tan Tieniu	3	16	69	HubauxJean-Pierre	8	18	72
Wattenhofer Roger	3	15	68	Lee Wenke	8	18	72
Kanade Takeo	4	30	131	Blaauw David	8	18	72
S.Philip	4	25	126	Sahai Amit	8	17	63
Giannakis Georgios	4	24	117	Prade Henri	9	32	111
Zhang Hong Jiang	4	22	110	Vetterli Martin	9	27	96
F.Ian	4	18	96	Kumar Vipin	9	24	95
WuJie	4	17	73	Deb Kalyanmoy	9	23	93
Sukhatme Gaurav	4	17	73	Benini Luca	9	20	85
Vasilakos Athanasios	4	16	67	McCallum Andrew	9	19	84
Cao Guohong	4	15	61	Kumar Ravi	9	18	69
Garcia-MolinaHector	5	29	125	LiXiang-Yang	9	17	57
Towsley Don	5	27	117	Demaine Erik	9	17	57
Culler David	5	24	113	H.Christos	10	34	110
Jennings Nick	5	22	107	Yager Ronald	10	33	101
Halevy Alon	5	21	94	Sangiovanni-VincentelliAlberto	10	32	99
Horvitz Eric	5	20	92	Agrawal Rakesh	10	25	95
J.Alexander	5	18	90	A.Thomas	10	24	93
Abdelzaher Tarek	5	17	72	Bellare Mihir	10	23	90
Fedkiw Ronald	5	16	60	Dorigo Marco	10	21	85
Poggio Tomaso	6	34	121	Karger David	10	20	84
E.Geoffrey	6	31	117	Friedman Nir	10	19	79
Pentland Alex	6	28	112	A.Carlos	10	18	67
VanLuc	6	22	104	D.Jeffrey	11	40	104
ChangShih-Fu	6	21	91	Shneiderman Ben	11	35	103
Szalay Alex	11	26	95	Mitzenmacher Michael	14	19	66
Sheth Amit	11	25	90	Reichert Manfred	14	18	60
Shah Mubarak	11	23	86	Davis Larry	15	36	93

Table A1. *Cont.*

Author	Rank	RP	H-Index	Author	Rank	RP	*h*-Index
Jiang Tao	11	22	85	Ayache Nicholas	15	29	90
Wooldridge Michael	11	21	81	Finin Tim	15	28	86
Gross Markus	11	20	77	Bertino Elisa	15	27	83
Domingos Pedro	11	19	75	Joaquin Jose	15	24	80
H.Jason	11	18	66	Mukher jee Biswanath	15	23	79
Suri Subhash	11	18	66	Vahdat Amin	15	22	74
Zadeh Lotfi	12	40	100	J.Michael	15	22	74
H.Gene	12	38	99	Joshi Anupam	15	21	71
E.David	12	27	95	K. Sajal	15	20	69
Widom Jennifer	12	26	91	Vaidya Nitin	15	20	69
ZhangLixia	12	25	88	Thiele Lothar	15	20	69
Dumais Susan	12	23	84	Pappas George	15	19	65
Schulzrinne Henning	12	22	81	Kraut Robert	16	29	88
Freeman William	12	22	81	Pedrycz Witold	16	28	83
Bengio Yoshua	12	21	76	Abadi Martin	16	27	81
Ray LiuK.J.	12	20	75	Hendler James	16	25	80
Wagner David	12	19	72	Roy Kaushik	16	23	76
Lu Songwu	12	18	63	Rus Daniela	16	21	70
W.Bruce	13	30	93	Handley Mark	16	21	70
Jajodia Sushil	13	27	92	Qiao Chunming	16	20	66
Anderson Thomas	13	26	86	Pollefeys Marc	16	19	63
Unser Michael	13	24	83	Y.Moshe	17	33	88
Manocha Dinesh	13	23	81	Doyle John	17	31	87
Perona Pietro	13	23	81	S.Kishor	17	31	87
Darrell Trevor	13	23	81	Alon Noga	17	29	85
Tsudik Gene	13	22	76	L.Ronald	17	29	85
Pevzner Pavel	13	22	76	Sontag Eduardo	17	28	82
Karypis George	13	22	76	C.Lee	17	26	79
Nahrstedt Klara	13	21	75	Taylor Chris	17	24	74
Yao Xin	13	21	75	S.Theodore	17	23	73
Diot Christophe	13	20	74	Reiter Michael	17	21	69
Goble Carole	13	19	69	Herrera Enrique	17	20	65
Liu Huan	13	19	69	Belongie Serge	17	19	62

Table A1. *Cont.*

Author	Rank	RP	H-Index	Author	Rank	RP	*h*-Index
Tse David	13	19	69	von John	18	40	87
Alouini Mohamed-Slim	13	18	61	Yannakakis Mihalis	18	35	86
M.John	14	33	93	A.David	18	30	84
Faugeras Olivier	14	30	91	Hebert Martial	18	30	84
Chellappa Rama	14	28	90	Dally William	18	29	80
De Giovanni	14	27	87	Blake Andrew	18	28	79
Sycara Katia	14	25	83	Baldi Pierre	18	26	77
Franklin Michael	14	24	81	Greenberg Saul	18	26	77
Rost Burkhard	14	23	80	S.Daniel	18	26	77
Crowcroft Jon	14	22	75	Alur Rajeev	18	25	74
McKeown Nick	14	20	73	B.Andrew	18	24	73
Suciu Dan	14	20	73	Fua Pascal	18	24	73
Varghese George	18	23	70	Rothermel Gregg	21	21	60
Yao Yiyu	18	23	70	Gray Jim	22	39	79
LeJean-Yves	18	23	70	Y.Joseph	22	32	78
PedramMassoud	18	23	70	BezdekJames	22	31	76
Savage Stefan	18	22	69	Kiesler Sara	22	31	76
Sandholm Tuomas	18	22	69	Terzopoulos Demetri	22	29	74
D.Gregory	18	22	69	Lenzerini Maurizio	22	27	70
Leymann Frank	18	21	65	J.Haim	22	27	70
Jha Somesh	18	21	65	Peterson Larry	22	26	69
Rogaway Philip	18	21	65	Shasha Dennis	22	26	69
R.John	18	21	65	Agrawal Divyakant	22	26	69
Shenoy Prashant	18	20	63	Baeza-YatesRicardo	22	25	68
Canetti Ran	18	20	63	C.JayC	22	24	67
Gunopulos Dimitrios	18	19	61	Stolcke Andreas	22	23	65
Pearl Judea	19	30	83	L.Michael	22	23	65
Ramakrishnan Raghu	19	29	78	Alonso Gustavo	22	22	62
Waibel Alex	19	28	77	S.B.	22	22	62
Li Kai	19	27	76	V.S.Laks	22	21	59
EtzioniOren	19	26	75	S.David	23	37	78
Cohen-OrDaniel	19	25	73	Magnenat-ThalmannNadia	23	32	75
Metaxas Dimitris	19	24	70	Kaufman Arie	23	29	72

Table A1. *Cont.*

Author	Rank	RP	H-Index	Author	Rank	RP	*h*-Index
Veloso Manuela	19	24	70	Devadas Srinivas	23	27	69
Smyth Padhraic	19	22	68	Cipolla Roberto	23	25	67
Kacprzyk Janusz	19	21	64	Salesin David	23	24	65
Schaffer Alejandro	19	21	64	Kotz David	23	23	63
Voelker Geoffrey	19	20	62	Druschel Peter	23	22	61
Decker Stefan	19	19	59	L.Olvi	24	40	78
Norman Don	20	40	83	C.Fernando	24	33	74
Bertsekas Dimitri	20	31	82	S.Andrew	24	30	71
Abiteboul Serge	20	29	77	Kautz Henry	24	28	69
Hanrahan Pat	20	28	76	Dill David	24	27	68
A.Edward	20	27	75	H.Mostafa	24	26	67
Cong Jason	20	25	70	Gropp William	24	25	65
Campbell Andrew	20	23	68	Ostrovsky Rafail	24	24	62
C.Ming	20	22	67	Altman Eitan	24	24	62
Zorzi Michele	20	21	61	Smyth Barry	24	22	59
Mylopoulos John	21	33	80	Crovella Mark	24	22	59
Thalmann Daniel	21	32	79	Newell Allen	25	40	75
Adeli Hojjat	21	30	76	Samet Hanan	25	36	73
Myers Brad	21	30	76	Harel David	25	33	72
Smith Barry	21	28	74	Mitchell Tom	25	32	71
Witten Ian	21	26	70	Yuille Alan	25	30	70
K.Sankar	21	25	69	D. Hill Mark	25	30	70
Sandhu Ravi	21	25	69	Stolfo Salvatore	25	30	70
J.Ingemar	21	24	68	G.Kim	25	29	68
Stojmenovic Ivan	21	23	67	Gottlob Georg	25	28	67
Cootes Tim	21	23	67	Haralick Robert	25	27	66
Anderson Ross	21	22	66	Nisan Noam	25	26	64
van Frank	25	25	62	Shadbolt Nigel	28	25	59
W.William	25	24	61	Ishibuchi Hisao	28	24	58
Rogers Yvonne	25	22	58	Rastogi Rajeev	28	24	58
Fagin Ronald	26	38	73	Gelenbe Erol	29	39	66
W.Thomas	26	34	70	H.Russell	29	33	65
Vitter Jeffrey	26	30	69	Reif John	29	33	65
Mooney Raymond	26	30	69	Salton Gerard	29	32	64
Cohen Michael	26	29	67	Dietterich Thomas	29	30	63
Canny John	26	29	67	Kramer Jeff	29	29	61
Burns Alan	26	28	66	Bajaj Chandrajit	29	29	61
Deriche Rachid	26	27	65	Aiken Alex	29	27	60

Table A1. *Cont.*

Author	Rank	RP	H-Index	Author	Rank	RP	*h*-Index
W.Wen-Mei	26	26	62	Wiederhold Gio	29	26	59
Keutzer Kurt	26	26	62	Dasgupta Dipankar	29	24	52
Pazzani Michael	26	26	62	Wilks Yorick	30	40	66
Blum Avrim	26	25	61	Turner Jonathan	30	31	63
Nejdl Wolfgang	26	24	60	Elmagarmid Ahmed	30	28	59
de Maarten	26	22	57	Motta Enrico	30	26	58
Ceri Stefano	27	32	68	Herman Gabor	31	40	65
Levy Henry	27	30	67	F.James	31	32	62
Tambe Milind	27	28	65	Larus James	31	29	58
K.Pankaj	27	27	63	ChenMing-Syan	31	27	57
Knoblock Craig	27	27	63	LeeDer-Tsai	31	26	53
Fogel David	27	24	59	Reddy Sudhakar	32	35	62
Baruah Sanjoy	27	23	56	Beth Mary	32	30	58
Bobrow Daniel	28	40	67	I.Norman	33	37	62
Hennessy John	28	30	66	Dolev Danny	33	35	60
Ni Lionel	28	29	65	Padua David	33	33	58
Wadler Philip	28	28	64	Nicolau Alex	33	31	56
Peleg David	28	28	64	V. Aho Alfred	34	40	62
P.Michael	28	27	62	Sifakis Joseph	34	32	56
Malik Sharad	28	25	59	A.Edward	35	32	55

Table A2. Indicators and Pareto ranks for the university data. Indicators are article, citation, Total Document (TD), Article Impact Total (AIT), Citation Impact Total (CIT), International Collaboration (IC), and the research period (RP).

University	Rank	Article	Citation	TD	AIT	CIT	IC	RP
Harvard University	1	126	126	60	108	90	90	380
University of Toronto	1	125	125	59	105.21	69.32	89	189
Stanford University	1	112.36	124.36	49.4	102.94	75.04	70.32	131
Johns Hopkins University	1	113.67	122.23	52.63	99.61	70.65	71.43	140
University of California Los Angeles	1	107.16	114.03	49.09	96.08	67.17	68.06	97
University of California San Diego	1	98.67	105.73	45.03	90.89	65.07	65.29	56
University of California Berkeley	2	105.06	117.51	44.83	103.2	74.81	71.13	148
Imperial College London	2	102.43	103.35	47.03	88.11	61.83	76.89	109
KU Leuven	2	98.43	94.22	44.4	83.16	56.85	76.42	48
Pierre & Marie Curie University - Paris 6	2	99.24	94.84	41.98	83.2	58.1	72.36	45
University of Oxford	2	115.22	119.72	51.44	103.96	72.49	85.11	920
University College London	2	116.44	113.55	54.13	97.6	65.34	84.34	190

Table A2. *Cont.*

University	Rank	Article	Citation	TD	AIT	CIT	IC	RP
University of Washington Seattle	2	108.58	116.21	48.95	97.19	68.28	67.17	155
Massachusetts Institute of Technology (MIT)	3	98.63	121.19	42.3	107	89	67.57	155
University of British Columbia	3	99.67	97.32	45.22	85.14	59.24	71.99	108
National University of Singapore	3	98.37	93.31	42.62	80.8	55.57	72.93	36
University of Cambridge	3	107.79	114.24	48.42	99.83	70.45	81.35	807
University of Michigan	3	114.84	113.74	51.62	97.45	64.64	68.39	199
University of Tokyo	3	108.06	101.7	46.83	87.92	57.8	66.55	139
Zhejiang University	4	111.19	89.26	44.39	77.61	51.52	60.98	119
Tsinghua University	4	107.94	89.06	41.91	79.41	53.07	60.46	105
Universidade de Sao Paulo	4	109.85	83.6	47.45	73.44	49.63	67.18	82
Seoul National University	4	102.76	89.17	44.18	75.39	51.38	59.98	70
Nanyang Technological University	4	88.18	86.44	37.67	75.84	54.57	64.08	25
University of Pennsylvania	4	105.54	113.05	49.83	93.83	66.55	63.03	276
University of Chicago	4	94.78	103.18	43.22	89.78	65.95	62.39	126
University of California San Francisco	4	93.26	107.72	45.4	85.02	65.38	60.39	143
Cornell University	4	97.05	100.49	44.67	85.3	60.8	63.24	151
University of Sydney	4	101.35	93.57	46.99	81.14	55.77	70.05	166
Monash University	4	95.25	86.58	42.4	73.35	51.78	64.32	58
Columbia University	4	103.52	109.56	47.37	93.12	66.28	66.84	262
Duke University	4	96.65	102.58	45.32	85.08	61.46	61.91	178
Shanghai Jiao Tong University	5	112.85	87.81	43.93	75.74	50.9	60.55	120
University of Melbourne	5	99.48	92.8	44.23	80.4	55.86	67.58	163
University of Queensland	5	98.53	90.06	42.98	77	53.44	67.19	107
University of California Davis	5	93.06	90.56	42.42	80.29	56.63	61.71	111
Free University of Berlin	5	88.74	86.87	41.62	71.8	52.31	62.58	68
University of Copenhagen	5	103.59	100.53	45.35	85.42	60.5	75.63	537
University of Minnesota Twin Cities	5	100.35	96.34	45.2	84.67	57.89	62	165
Central South University	6	86.42	71.48	36.15	61.6	46.73	50.01	16
Peking University	6	106.48	90.88	42.87	79.04	53.52	63	118
University of Colorado Boulder	6	95.39	95.59	42.51	81.87	59.08	59.03	140
Ohio State University	6	97.46	91.51	43.97	82.1	57.84	60.06	146
University of Florida	6	93.55	88.3	42.92	77.72	54.49	60.9	111
Aarhus University	6	90	85.21	40.03	72.48	52.05	65.48	88
University of Wisconsin Madison	6	96.46	95.44	43.5	86.34	60.62	60.17	168
University of Pittsburgh	6	94.27	98.62	45.51	81.7	58.97	58.91	229

Table A2. *Cont.*

University	Rank	Article	Citation	TD	AIT	CIT	IC	RP
Yale University	6	96.84	103.83	45	88.02	64.44	62.73	315
Swiss Federal Institute of Technology Zurich	7	91.5	90.14	39.43	80.86	57.2	69.92	162
California Institute of Technology	7	80.64	91.01	35.94	80.84	64.69	58.91	125
University of Paris Diderot - Paris VII	7	82.43	84.92	37.03	73.33	56.17	60.31	46
Radboud University Nijmegen	7	83.98	82.98	38.7	72.31	53.82	61.26	93
McGill University	7	95.02	91.31	43.21	79.8	56.3	67.91	195
Kyoto University	7	94.92	87.64	42.22	75.7	52.31	58.98	119
University of New South Wales	7	91.21	83.93	40.66	72.61	51.45	63.02	67
University of North Carolina Chapel Hill	7	93.35	96.13	43.42	79.93	58.08	57.65	227
Erasmus University Rotterdam	8	82.84	85.67	39.48	71.33	53.86	59.63	103
University of Calgary	8	80.47	77.12	37.76	65	48.87	56.69	50
Maastricht University	8	77.35	74.82	35.99	63.26	48.58	55.98	40
University of California Santa Cruz	8	68.71	76.1	32.27	66.59	57	49.98	51
Northwestern University	8	92.54	94.9	42.41	80.53	58.41	57.42	165
Penn State University	8	94.46	91.75	42.03	79.6	55.67	60.55	161
University of Texas Austin	8	88.6	88.4	39.15	78.45	57.02	57.21	135
University of Alberta	8	89.92	82.62	40.87	72.13	51.38	62.21	108
Ecole Polytechnique Federale de Lausanne	8	78.45	81.86	35	71.47	54.97	59	51
University of Bristol	9	81.61	79.97	37.38	71.3	52.96	57.94	85
University of Paris Descartes - Paris V	9	78.7	78.14	36.4	64.68	49.81	55.65	45
University of Manchester	9	92.82	88.76	42.55	78.18	55.6	65.72	192
Washington University (WUSTL)	9	85.81	95.32	40.77	77.39	60.3	54.55	163
Fudan University	9	96.36	84.67	39.85	71.27	51.04	56.62	111
University of Southern California	9	86.45	86.57	39.75	73.86	54.66	56.58	136
VU University Amsterdam	9	86.49	84.65	39.82	72.01	52.62	62.11	136
University of Utrecht	9	92.3	92.51	42.21	78.61	56.05	66.52	380
University of Edinburgh	9	87.11	90.17	40.58	78.94	58.37	63.87	433
National Taiwan University	9	90.25	81.46	40.39	71.78	50.26	56.72	88
University of California Irvine	10	79.5	82.59	37.07	72.39	54.64	54.56	109
University of Claude Bernard - Lyon 1	10	77.39	75.36	35.21	66.46	50.37	55.59	45
Kings College London	10	88.56	87.8	42.28	75.5	55.64	63.08	187
University of Zurich	10	86.5	86.17	39.27	75.24	56.02	65.28	183
Vanderbilt University	10	84.94	88	39.85	74.73	56.02	54.13	143
University of Arizona	10	83.28	82.38	38.22	73.32	53.9	56.68	131
Sun Yat Sen University	10	93.75	79.59	38.93	68.85	49.69	53.61	92

Table A2. *Cont.*

University	Rank	Article	Citation	TD	AIT	CIT	IC	RP
University of Science & Technology of China	10	87.03	79.64	36.29	70.31	50.91	53.34	58
University of Hamburg	10	80.76	77.61	36.9	69.02	52.67	57.03	97
Tel Aviv University	10	83.45	77	38.05	67.76	49.97	57.41	60
University of Barcelona	10	93.33	92.11	42.39	77	55.09	66.57	566
Ruprecht Karl University Heidelberg	10	88.26	91.02	41.16	77.82	57.81	63.37	630
Karolinska Institutet	10	90.45	90.45	41.42	73.53	54.38	67.8	206
University of Munich	10	88.4	89.24	40.73	76.67	56.6	64.03	544
Osaka University	10	87.89	81.47	39.53	69.8	50.18	54.72	85
University of Milan	10	82.84	78.7	38.18	68.29	51.19	56.66	92
University of Illinois Urbana-Champaign	11	88.61	83.81	39.34	75.86	53.9	57.68	149
Nanjing University	11	92.16	79.86	37.74	70.4	50.36	53.5	101
University of Geneva	11	78.31	80.67	36.53	71.01	54.96	59.55	140
University of Birmingham	11	79.7	78.21	38.05	69.02	52.11	56.73	116
Autonomous University of Barcelona	11	80.98	76.79	36.54	67.87	50.52	57.14	48
Universite Toulouse III - Paul Sabatier	11	77.42	75.94	34.94	65.12	49.64	56.82	47
University of Alabama Birmingham	11	75.88	78.19	36.86	64.02	49.9	50.24	47
Ghent University	11	92.24	83.31	40.58	74.82	52.29	67.23	199
New York University	11	88.57	86.84	41.02	76.06	55.34	56.61	185
Humboldt University of Berlin	11	87.79	86.45	41.06	73.29	53.69	61.82	205
Boston University	11	82.06	86.78	38.39	76.7	58.5	55.3	177
University of Montreal	11	84.99	81.52	39.57	70.85	51.58	60.52	138
Tohoku University	11	86.88	80.11	39.29	68.53	49.14	56.83	105
Universidade de Lisboa	11	84.96	75.09	37.75	67.52	48.99	60.59	105
University of Amsterdam	11	89.6	87.32	40.78	75.05	54.47	63.83	384
King Abdulaziz University	12	83.13	71.26	34.2	62.39	47.72	61.55	49
University of Maryland College Park	12	85.53	85.24	37.8	77.5	56.03	57.62	160
Huazhong University of Science & Technology	12	94.22	76.78	38.13	67.75	48.51	52.97	109
University of California Santa Barbara	12	74.07	79.24	34.15	70.92	56.17	52.51	125
King Saud University	12	81.61	70.42	35.17	60.55	46.49	61	59
Technical University of Munich	12	85.41	82.83	38.02	70.76	52.07	60.24	148
Australian National University	12	80.86	76.09	36.24	66.79	49.53	57.88	70
Jilin University	12	90.4	75.46	37.77	64.59	47.77	50.8	70
University of Groningen	12	88.8	87.97	40.77	74.11	53.71	63.99	402
University of Helsinki	13	85.98	84.46	38.79	73.96	54.45	63.63	376
Emory University	13	85.31	88.39	41.09	71.91	54.01	54.36	180

Table A2. *Cont.*

University	Rank	Article	Citation	TD	AIT	CIT	IC	RP
University of Oslo	13	84.56	81.86	38.4	71.61	53.18	61.9	205
Princeton University	13	79.62	84.19	35.91	76.18	58.53	55.73	270
Shandong University	13	92.53	75.48	37.9	66.61	48.45	51.74	115
University of Leeds	13	79.38	76.98	37.34	66.19	49.77	56.86	112
Newcastle University - UK	13	75.18	74.08	36.15	63.05	48.62	53.74	53
Sapienza University Rome	13	90.05	81.21	40.4	74.67	53.66	60.14	713
University of Hong Kong	13	81.72	77.6	36.83	66.37	49.24	54.98	105
Harbin Institute of Technology	13	88.78	73.47	37.14	65.59	47.69	51.92	96
Universidad Nacional Autonoma de Mexico	13	82.93	71.29	36.55	61.79	46.56	56.98	106
University of Miami	13	75.35	77.57	36.29	64.56	50.1	51.28	91
Purdue University	14	84.43	78.86	37.5	71.8	51.79	55.03	140
McMaster University	14	80.98	79.62	38.05	67.28	50.46	57.26	129
Sichuan University	14	91.37	74.89	38.2	63.79	47.25	50.46	120
Nagoya University	14	79.99	74.5	36.4	65.02	48.7	51.87	77
University of Gothenburg	14	77.19	74.76	35.24	63.31	48.54	55.41	62
Leiden University	14	83.95	85.71	38.7	72.45	54.93	60.88	441
Lund University	14	85.58	82.77	38.57	72.11	52.93	63.82	350
Hebrew University of Jerusalem	14	77.35	75.99	35.91	64.76	49.23	55.37	98
Wageningen University & Research Center	14	76.89	75.2	34.63	65.23	49.34	56.74	98
Georgia Institute of Technology	15	80.78	78.99	35.89	69.09	51.11	54.8	131
University of Waterloo	15	78.08	72.61	34.88	64.28	48.17	54.96	59
Rutgers State University	15	82.92	82.39	38.89	73.02	53.17	56.11	250
Texas A & M University College Station	15	85.21	78.81	37.8	70.59	50.83	57.22	163
University of Southampton	15	82.64	78.11	37.56	69.57	51.27	59.62	147
Michigan State University	15	83.49	79.5	37.73	70.72	51.52	55.15	161
University of Sheffield	15	78.55	75.56	36.89	67.27	50.39	55.73	111
University of Illinois Chicago	15	77.49	74.21	36.76	65.53	49.57	50.96	103
University of Paris Sud - Paris XI	15	83.29	82.86	37.31	73.7	54.24	61.5	759
Uppsala University	15	84.96	82.76	37.95	70.97	52.49	62.68	539
University of Aix-Marseille	15	85.64	81.77	38.19	71.74	52.21	61.44	607
University of Utah	16	82.83	81.24	38.29	68.1	51.19	53.08	166
University of Nottingham	16	80.15	78.05	37.97	67.01	50.12	56.48	135
University of Bonn	16	77.93	78.15	36.02	68.19	52.04	57.07	198
Yonsei University	16	87.79	76.2	38.48	65.27	47.85	52.65	131
Xian Jiaotong University	16	88.95	72.43	36.56	63.84	47.52	51.96	120
Universidade do Porto	16	79.21	72.6	36	62.61	47.42	56.16	105
Goethe University Frankfurt	16	74.56	75.63	35.37	63.9	49.5	54.17	102

Table A2. *Cont.*

University	Rank	Article	Citation	TD	AIT	CIT	IC	RP
University of Padua	16	85.18	81.27	38.58	71.73	53.36	59.63	794
University of Western Australia	16	84.57	80.2	38.33	67.65	50.13	60.08	457
Universite Grenoble Alpes (UGA)	16	77.79	79.09	36.13	71.06	52.39	60.35	474
University of Bern	17	79.33	76.3	36.17	67.17	51.5	58.97	182
University of Virginia	17	78.15	78.2	36.72	68.55	52.23	52.2	197
Arizona State University	17	80.2	77.56	36.18	67.63	50.64	52.32	131
University of Iowa	17	78.68	76.15	36.98	68.19	51.93	51.01	169
Korea University	17	82.74	74.18	36.72	65.69	48.43	52.28	111
Cardiff University	17	75	76.04	35.9	65.22	50.76	54.59	133
University of Cologne	17	76.7	73.92	35.31	62.77	48.49	54.45	97
Charite Medical University of Berlin	17	78.46	80.17	37.97	65.5	50.48	55.65	306
University of Liverpool	17	77.49	74.92	36.09	65.56	50	56.24	135
Kyushu University	17	80.46	73.69	36.6	63.72	47.82	51.47	105
Tongji University	17	84.92	71.28	35.94	61.55	46.65	51.38	109
Hokkaido University	17	79.61	73.24	36.18	62.39	47.2	51.53	98
University of Bologna	17	81.81	76.2	37.47	69.83	51.99	56.52	928
Universite de Toulouse	17	81.97	77.85	36.5	67.4	49.97	59.67	787
University of Glasgow	17	77.61	77.37	37	68.44	51.98	56.16	565
Stockholm University	18	75.64	74.11	33.97	65.57	50.76	55.41	138
Brown University	18	77.68	78.44	36.28	68.39	52.87	51.02	252
Dresden University of Technology	18	78.34	76.02	35.85	66.66	50.43	55.25	188
RWTH Aachen University	18	77.44	74.74	35.37	65.96	50.2	54.39	146
University of Rochester	18	74.66	76.75	35.64	66.57	51.31	51.18	166
Wuhan University	18	85.16	73.3	35.54	63.01	47.56	50.48	123
Eberhard Karls University of Tubingen	18	78.97	78.74	36.83	66.24	50.28	57.19	539
University of Basel	18	77.08	77.77	35.86	65.65	50.75	58.36	556
Technical University of Denmark	19	78.02	75.91	34.54	66.6	50.24	56.59	187
University of Gottingen	19	77.24	75.75	35.5	66.69	50.57	55.63	282
University of Ottawa	19	79.59	75.55	37.19	64.48	48.61	54.72	168
Case Western Reserve University	19	76.74	78.47	36.39	65.74	51.09	50.86	190
Western University (University of Western Ontario)	19	78.89	74	36.79	63.15	47.77	54.05	138
University of Auckland	19	76.53	71.95	35.01	63.94	48.93	54.97	133
Sungkyunkwan University	19	85.09	76.39	37	66.37	49.54	51.96	618
University of Freiburg	19	76.35	76.61	35.91	67.04	51.55	55.89	559
University of Erlangen Nuremberg	20	79.19	76.92	36.16	65.15	49.36	55.93	273
University of Adelaide	20	79.41	73.54	35.7	63.9	48.62	54.39	142
Lomonosov Moscow State University	20	82.26	69.62	36.05	64.56	48.18	54.81	261
North Carolina State University	20	79.13	73.4	35.37	64.17	48.05	51.84	129

Table A2. *Cont.*

University	Rank	Article	Citation	TD	AIT	CIT	IC	RP
Universite de Montpellier	20	78.34	76.42	36.02	65.05	49.07	58.46	727
Karlsruhe Institute of Technology	20	77.15	73.49	34.36	65.91	49.88	55.64	191
University of Munster	21	75.94	76.75	35.61	64.51	49.82	54.1	236
Queen Mary University London	21	73.49	74.2	35.02	64.12	50.58	52.71	231
Charles University Prague	21	79.55	73.28	36.4	65.57	49.64	56.59	668
University of Naples Federico II	22	79.04	73.63	35.99	66.4	50.47	53.7	792
University of Turin	23	76.97	74.11	35.51	64.7	49.94	53.26	612
Johannes Gutenberg University of Mainz	23	74.47	73.39	34.66	64.26	49.86	53.68	539

Table A3. Indicators and Pareto ranks for the country data. Indicators are documents, Citable Documents (CI-DO), citations, Citations Per Document (CPD), and *h*-index.

Country	Rank	Documents	CI-DO	Citations	CPD	*h*-Index
United States	1	9,360,233	8,456,050	202,750,565	21.66	1783
Netherlands	2	746,289	682,627	16,594,528	22.24	752
United Kingdom	2	2,624,530	2,272,675	50,790,508	19.35	1099
Switzerland	3	541,846	501,917	12,592,003	23.24	744
China	3	4,076,414	4,017,123	24,175,067	5.93	563
Germany	3	2,365,108	2,207,765	40,951,616	17.31	961
Canada	3	1,339,471	1,227,622	25,677,205	19.17	862
Panama	4	5129	4830	137,585	26.82	142
Sweden	4	503,889	471,036	10,832,336	21.5	666
Denmark	4	290,994	269,364	6,405,076	22.01	558
Iceland	4	15,625	14,353	357,678	22.89	218
Japan	4	2,212,636	2,133,326	30,436,114	13.76	797
France	4	1,684,479	1,582,197	28,329,815	16.82	878
Gambia	5	2004	1859	54,925	27.41	99
Israel	5	295,747	274,748	5,826,878	19.7	536
Belgium	5	407,993	378,807	7,801,077	19.12	593
Italy	5	1,318,466	1,217,804	20,893,655	15.85	766
Australia	5	995,114	894,315	16,321,650	16.4	709
Bermuda	6	633	590	21,884	34.57	73
Finland	6	257,159	242,853	4,940,153	19.21	479
Spain	6	1,045,796	966,710	14,811,902	14.16	648
Montserrat	7	95	93	2282	24.02	27
Austria	7	295,668	273,467	5,052,810	17.09	487
India	7	1,140,717	1,072,927	8,458,373	7.41	426
South Korea	7	824,839	801,077	8,482,515	10.28	476

Table A3. *Cont.*

Country	Rank	Documents	CI-DO	Citations	CPD	*h*-Index
Taiwan	7	532,534	516,171	5,622,744	10.56	363
Faroe Islands	8	510	472	10,105	19.81	48
United States Minor Outlying Islands	8	30	29	710	23.67	11
Norway	8	229,276	209,259	3,951,661	17.24	439
Brazil	8	669,280	639,527	5,998,898	8.96	412
Guinea-Bissau	9	458	421	9357	20.43	50
Puerto Rico	9	13,841	13,293	248,888	17.98	166
Hong Kong	9	219,177	206,011	3,494,244	15.94	392
Greece	9	246,202	226,914	3,186,313	12.94	354
Russian Federation	9	770,491	755,186	4,907,109	6.37	421
Poland	9	475,693	460,979	4,083,631	8.58	401
Tokelau	10	2	1	43	21.5	1
Monaco	10	1586	1449	29,705	18.73	76
New Zealand	10	180,340	162,720	2,940,051	16.3	387
Singapore	10	215,553	202,089	3,135,524	14.55	392
Turkey	10	434,806	407,064	3,509,424	8.07	296
French Southern Territories	11	5	5	97	19.4	5
Bolivia	11	3569	3387	61,076	17.11	88
Ireland	11	150,552	135,523	2,382,077	15.82	364
Czech Republic	11	237,910	230,048	2,204,922	9.27	322
Mexico	11	232,828	221,611	2,305,554	9.9	316
Portugal	11	214,838	201,562	2,544,577	11.84	334
Argentina	11	159,172	150,927	1,965,624	12.35	300
Costa Rica	12	9177	8612	148,475	16.18	137
Gabon	12	2048	1936	34,704	16.95	80
Hungary	12	147,901	140,910	1,914,820	12.95	329
Kenya	12	24,458	22,347	379,560	15.52	179
South Africa	12	188,104	172,424	2,125,927	11.3	320
Iran	12	333,474	323,299	1,954,324	5.86	199
Seychelles	13	482	453	8579	17.8	44
North Korea	13	2384	2329	38,622	16.2	80
New Caledonia	13	2122	2041	34,753	16.38	73
Estonia	13	28,660	27,323	381,206	13.3	185
Chile	13	101,841	97,250	1,203,308	11.82	257
Uganda	13	11,528	10,599	171,367	14.87	128
Thailand	13	123,410	117,565	1,182,686	9.58	236
Egypt	13	137,350	133,147	1,009,954	7.35	184

Table A3. *Cont.*

Country	Rank	Documents	CI-DO	Citations	CPD	*h*-Index
Malaysia	13	181,251	175,146	888,277	4.9	190
Saint Lucia	14	99	85	1774	17.92	17
Netherlands Antilles	14	435	397	7662	17.61	44
Martinique	14	653	598	10,737	16.44	39
Philippines	14	20,326	18,658	265,737	13.07	163
Tanzania	14	11,964	11,140	170,144	14.22	122
Slovenia	14	71,408	68,494	725,498	10.16	204
Saudi Arabia	14	111,117	106,187	748,069	6.73	195
Slovakia	14	80,765	78,484	653,526	8.09	195
Romania	14	141,731	138,041	752,219	5.31	187
Malawi	15	4952	4520	77,829	15.72	104
Peru	15	14,434	13,201	192,443	13.33	154
Uruguay	15	13,702	12,971	186,793	13.63	132
Bulgaria	15	59,384	57,590	523,844	8.82	184
Venezuela	15	33,780	32,445	321,006	9.5	166
Ukraine	15	145,332	142,812	732,429	5.04	188
Croatia	15	79,154	76,097	548,687	6.93	194
French Guiana	16	956	898	15,573	16.29	56
Mozambique	16	2382	2193	37,433	15.71	73
Ecuador	16	7942	7440	96,119	12.1	111
Zimbabwe	16	7243	6691	94,533	13.05	99
Zambia	16	3992	3623	56,481	14.15	92
Cyprus	16	17,072	15,552	172,117	10.08	127
Pakistan	16	94,285	90,034	546,210	5.79	166
Colombia	16	60,402	57,407	468,135	7.75	186
Viet Nam	16	29,238	27,989	253,661	8.68	142
Lebanon	16	20,815	19,040	186,558	8.96	138
Virgin Islands (British)	17	121	111	2047	16.92	20
Mali	17	2490	2353	36,254	14.56	75
Armenia	17	12,852	12,496	130,584	10.16	135
Nigeria	17	59,372	56,630	334,059	5.63	131
Tunisia	17	58,769	55,904	342,429	5.83	123
Indonesia	17	39,719	37,729	282,788	7.12	155
Lithuania	17	36,136	35,205	271,666	7.52	144
Kuwait	17	18,468	17,687	157,888	8.55	108
Hati	18	765	683	12,231	15.99	49
French Polynesia	18	1272	1207	19,523	15.35	58
Senegal	18	7220	6752	75,373	10.44	95

Table A3. *Cont.*

Country	Rank	Documents	CI-DO	Citations	CPD	*h*-Index
Cambodia	18	2558	2292	34,654	13.55	72
Sri Lanka	18	12,557	11,532	121,696	9.69	120
Morocco	18	40,737	38,371	279,731	6.87	129
Ethiopia	18	13,363	12,625	118,656	8.88	101
Bangladesh	18	30,612	29,157	227,447	7.43	134
Guam	19	788	727	12,222	15.51	55
Cte dIvoire	19	4842	4621	52,446	10.83	89
Madagascar	19	3207	3059	39,217	12.23	74
Papua New Guinea	19	2258	2133	31,119	13.78	71
Luxembourg	19	12,562	11,567	120,570	9.6	114
United Arab Emirates	19	31,366	29,259	210,873	6.72	130
Belarus	19	30,944	30,439	202,088	6.53	133
Jordan	19	28,234	27,369	201,400	7.13	112
Nicaragua	20	1301	1233	18,269	14.04	62
Greenland	20	977	941	14,484	14.82	48
Namibia	20	2303	2125	28,985	12.59	72
Guatemala	20	2281	2085	29,034	12.73	69
Ghana	20	11,543	10,578	111,205	9.63	105
Serbia	20	53,116	50,436	258,732	4.87	118
Algeria	20	42,456	41,544	215,922	5.09	106
Cuba	20	31,690	30,382	202,503	6.39	127
Latvia	20	16,350	15,851	119,627	7.32	112
Cameroon	21	11,128	10,513	108,649	9.76	94
Democratic Republic Congo	21	517	481	7641	14.78	43
Georgia	21	11,196	10,305	105,036	9.38	114
Oman	21	12,846	11,919	87,333	6.8	91
Palau	22	149	143	2238	15.02	26
Botswana	22	5107	4545	52,195	10.22	79
Barbados	22	1690	1416	20,879	12.35	64
Nepal	22	9133	8196	85,174	9.33	94
Qatar	22	13,438	12,524	71,382	5.31	86
Congo	23	3304	3069	34,559	10.46	72
Honduras	23	995	950	13,157	13.22	51
Guinea	23	597	552	8320	13.94	46
Jamaica	23	4750	4220	48,226	10.15	75
Niger	23	1623	1553	19,835	12.22	59
Sudan	23	6099	5792	50,784	8.33	70
Syrian Arab Republic	23	5744	5459	53,601	9.33	81
Macedonia	23	8522	8167	54,409	6.38	81

Table A3. *Cont.*

Country	Rank	Documents	CI-DO	Citations	CPD	*h*-Index
Laos	24	1802	1670	20,028	11.11	59
Belize	24	330	299	4734	14.35	38
Virgin Islands (U.S.)	24	215	204	3173	14.76	31
Mongolia	24	3319	3164	33,119	9.98	72
Paraguay	24	1454	1373	17,717	12.19	60
Moldova	24	5948	5828	46,522	7.82	80
Malta	24	4500	3980	40,668	9.04	83
Trinidad and Tobago	24	5037	4561	44,146	8.76	76
Sao Tome and Principe	25	47	45	695	14.79	15
Chad	25	382	363	5122	13.41	33
Guadeloupe	25	1435	1345	17,075	11.9	52
Benin	25	3851	3681	35,470	9.21	65
Kazakhstan	25	12,124	11,809	39,700	3.27	68
Iraq	25	11,605	11,042	39,145	3.37	59
Uzbekistan	25	9259	8997	46,900	5.07	68
Palestine	25	4506	4224	30,338	6.73	60
Central African Republic	26	538	500	6940	12.9	41
Fiji	26	2400	2188	22,836	9.52	56
Liechtenstein	26	1272	1172	14,339	11.27	55
Dominican Republic	26	1101	1029	12,965	11.78	51
Azerbaijan	26	9848	9620	40,070	4.07	64
Yemen	26	2776	2698	18,951	6.83	50
Macao	26	5157	4903	25,298	4.91	57
Bahrain	26	4657	4225	24,769	5.32	55
Falkland Islands (Malvinas)	27	358	341	4628	12.93	34
American Samoa	27	162	150	2127	13.13	22
Gibraltar	27	106	94	1451	13.69	19
Mauritius	27	2206	2035	17,629	7.99	54
Rwanda	27	1759	1554	15,356	8.73	54
Myanmar	27	1543	1458	13,764	8.92	51
Reunion	27	581	544	6605	11.37	38
Bosnia and Herzegovina	27	7054	6752	30,300	4.3	61
Brunei Darussalam	27	2440	2136	16,224	6.65	52
Albania	27	3172	3028	14,759	4.65	48
Solomon Islands	28	324	296	4125	12.73	33
Svalbard and Jan Mayen	28	20	18	283	14.15	8
Tonga	28	108	105	1408	13.04	21
Sierra Leone	28	590	529	5551	9.41	31

Table A3. *Cont.*

Country	Rank	Documents	CI-DO	Citations	CPD	*h*-Index
Kyrgyzstan	28	1486	1402	9918	6.67	45
El Salvador	28	1149	1061	9994	8.7	44
Swaziland	28	1091	988	9618	8.82	43
Eritrea	28	488	468	5260	10.78	35
Bahamas	28	399	365	4535	11.37	36
Libya	28	4160	4020	18,971	4.56	51
San Marino	29	191	181	2365	12.38	23
British Indian Ocean Territory	29	19	16	267	14.05	7
Guyana	29	530	485	4898	9.24	32
Togo	29	1470	1367	8850	6.02	39
Angola	29	715	680	5422	7.58	35
Mauritania	29	482	456	4762	9.88	32
Samoa	29	249	231	2734	10.98	27
Montenegro	29	2232	2153	7346	3.29	32
Saint Vincent and the Grenadines	30	40	38	518	12.95	11
Federated States of Micronesia	30	188	175	2144	11.4	24
Grenada	30	965	824	6286	6.51	33
Afghanistan	30	791	674	5800	7.33	36
Vanuatu	30	317	295	3142	9.91	27
Tajikistan	30	1244	1209	4728	3.8	29
Lesotho	30	459	425	3524	7.68	28
Burundi	30	421	392	3761	8.93	32
Suriname	31	293	276	2921	9.97	30
Bhutan	31	551	499	3249	5.9	27
Andorra	31	172	151	1786	10.38	21
Turkmenistan	31	296	286	2291	7.74	20
Cocos (Keeling) Islands	32	14	14	162	11.57	4
Tuvalu	32	25	24	284	11.36	8
Dominica	32	266	234	2007	7.55	23
Cayman Islands	32	231	210	1857	8.04	23
Maldives	32	206	194	1833	8.9	21
Equatorial Guinea	32	153	147	1587	10.37	20
Turks and Caicos Islands	33	45	45	475	10.56	13
Saint Kitts and Nevis	33	350	240	1866	5.33	21
Liberia	33	263	216	1934	7.35	21
Comoros	33	96	89	839	8.74	13
Marshall Islands	33	84	77	827	9.85	16
Northern Mariana Islands	33	68	66	680	10	14
Cook Islands	33	64	61	658	10.28	14

Table A3. *Cont.*

Country	Rank	Documents	CI-DO	Citations	CPD	*h*-Index
Cape Verde	34	199	194	1501	7.54	17
Djibouti	35	190	178	1206	6.35	18
Aruba	36	93	74	621	6.68	12
Somalia	36	115	97	685	5.96	15
Timor-Leste	37	125	102	628	5.02	13
Mayotte	37	74	72	416	5.62	10
Antigua and Barbuda	38	114	103	550	4.82	13
Anguilla	38	36	33	201	5.58	7
South Georgia and the South Sandwich Islands	39	7	5	42	6	2
Kiribati	39	33	28	184	5.58	8
Norfolk Island	40	20	20	114	5.7	7
Nauru	40	22	21	118	5.36	6
Vatican City State	41	25	16	121	4.84	6
Christmas Island	41	7	7	38	5.43	4
Saint Helena	42	15	15	69	4.6	5
Niue	43	16	13	25	1.56	2
Bouvet Island	43	6	4	29	4.83	2
Wallis and Futuna	43	15	13	60	4	4
Western Sahara	44	11	9	22	2	3
Heard Island and McDonald Islands	45	1	1	3	3	1
Saint Pierre and Miquelon	45	5	4	6	1.2	1
Pitcairn	46	3	1	4	1.33	1

Table A4. Pareto ranks and ranks from three sites.

University	Rank1	Rank2	Rank3	Pareto Rank
Massachusetts Institute of Technology	1	3	7	1
Stanford University	2	2	4	1
Harvard University	3	1	1	1
University of Cambridge	4	4	8	2
University of Oxford	6	5	3	2
University of Toronto	32	30	2	2
California Institute of Technology	5	11	59	3
University College London	7	31	5	3

Table A4. *Cont.*

University	Rank1	Rank2	Rank3	Pareto Rank
University of Chicago	10	8	20	3
Yale University	15	10	19	3
Johns Hopkins University	17	16	6	3
University of Pennsylvania	18	14	13	3
Columbia University	20	6	14	3
University of California, Berkeley (UCB)	28	7	9	3
Swiss Federal Institute of Technology	8	96	5	4
Imperial College London	9	35	15	4
Princeton University	11	9	93	4
Cornell University	16	12	25	4
University of Michigan	23	19	10	4
University of California, Los Angeles (UCLA)	31	15	12	4
University of Tokyo	34	13	18	4
Pennsylvania State University	95	14	13	4
National University of Singapore (NUS)	12	63	29	5
The University of Edinburgh	19	55	52	5
Duke University	25	29	24	5
Northwestern University	26	21	46	5
Kyoto University	37	20	60	5
University of California, San Diego (UCSD)	40	17	17	5
University of Washington	59	27	11	5
Nanyang Technological University	13	134	66	6
Tsinghua University	24	74	38	6
The University of Manchester	29	61	49	6
McGill University	30	42	35	6
Seoul National University	35	24	50	6
Peking University	39	60	33	6
The University of Melbourne	42	89	31	6
University of British Columbia	45	57	21	6
New York University	46	22	68	6
University of Wisconsin-Madison	53	25	30	6
University of Copenhagen	68	69	16	6
The University of Hong Kong	27	169	137	7
University of Bristol	41	129	102	7
Fudan University	43	192	74	7

Table A4. *Cont.*

University	Rank1	Rank2	Rank3	Pareto Rank
University of Sydney	46	95	27	7
Brown University	49	87	144	7
Carnegie Mellon University	58	67	247	7
Osaka University	63	48	101	7
University of Illinois at Urbana-Champaign	66	34	76	7
University of Texas at Austin	67	32	64	7
Ruprecht Karl University Heidelberg	72	82	51	7
University of North Carolina, Chapel Hill	78	38	43	7
Katholieke Universiteit Leuven	79	78	23	7
The Ohio State University	88	46	37	7
The Hong Kong University of Science and Technology	36	312	325	8
The University of New South Wales (UNSW Australia)	49	117	71	8
University of Queensland	51	99	41	8
Shanghai Jiao Tong University	61	166	39	8
National Taiwan University (NTU)	68	53	92	8
University of Zurich	80	93	65	8
University of California, Davis	85	49	47	8
Utrecht University	104	83	44	8
University of Warwick	51	280	208	9
Tokyo Institute of Technology	56	128	253	9
University of Amsterdam	57	111	61	9
Technical University of Munich	60	104	95	9
Monash University	65	143	57	9
Georgia Institute of Technology	71	86	125	9
Tohoku University	75	84	105	9
Boston University	89	62	79	9
University of Helsinki	91	107	72	9
Purdue University	92	56	109	9
University of Alberta	94	101	77	9
Washington University (WUSTL)	106	51	56	9
City University of Hong Kong	55	364	252	10
Delft University of Technology	62	255	210	10
University of Glasgow	63	132	130	10
Lund University	73	127	83	10
Rice University	90	114	292	10

Table A4. *Cont.*

University	Rank1	Rank2	Rank3	Pareto Rank
University of Geneva	95	80	103	10
Uppsala University	98	126	89	10
Leiden University	102	112	82	10
Lomonosov Moscow State University	108	77	177	10
Durham University	74	231	258	11
The University of Nottingham	75	139	127	11
University of Birmingham	82	158	119	11
University of Southampton	87	153	110	11
Royal Institute of Technology	97	131	206	11
The University of Western Australia	102	213	104	11
University of St Andrews	77	307	348	12
The University of Auckland	81	252	195	12
Pohang University of Science And Technology (POSTECH)	83	191	349	12
The University of Sheffield	84	172	147	12
University of Leeds	93	159	138	12
Korea University	98	141	162	12
University of Science and Technology of China	104	223	113	12
Universidad de Buenos Aires (UBA)	85	372	277	13
Trinity College Dublin	98	175	263	13
Karlsruhe Institute of Technology	101	215	172	13
Sungkyunkwan University (SKKU)	106	221	139	13
Technical University of Denmark	109	168	154	13

References

1. Khan, N.R.; Thompson, C.J.; Taylor, D.R.; Gabrick, K.S.; Choudhri, A.F.; Boop, F.R.; Klimo, P. Part II: Should the h-index be modified? An analysis of the m-quotient, contemporary h-index, authorship value, and impact factor. *World Neurosurg.* **2013**, *80*, 766–774. [CrossRef] [PubMed]
2. Baldock, C.; Ma, R.; Orton, C.G. The h index is the best measure of a scientist's research productivity. *Med. Phys.* **2009**, *36*, 1043–1045. [CrossRef]
3. Rezek, I.; McDonald, R.J.; Kallmes, D.F. Is the h-index predictive of greater NIH funding success among academic radiologists? *Acad. Radiol.* **2011**, *18*, 1337–1340. [CrossRef] [PubMed]
4. Waltman, L.; Calero-Medina, C.; Kosten, J.; Noyons, E.; Tijssen, R.J.; Eck, N.J.; Leeuwen, T.N.; Raan, A.F.; Visser, M.S.; Wouters, P. The Leiden Ranking 2011/2012: Data collection, indicators, and interpretation. *J. Am. Soc. Inf. Sci. Technol.* **2012**, *63*, 2419–2432. [CrossRef]
5. Patel, V.M.; Ashrafian, H.; Ahmed, K.; Arora, S.; Jiwan, S.; Nicholson, J.K.; Darzi, A.; Athanasiou, T. How has healthcare research performance been assessed? A systematic review. *J. R. Soc. Med.* **2011**, *104*, 251–261. [CrossRef]
6. Hirsch, J.E. An index to quantify an individual's scientific research output. *Proc. Natl. Acad. Sci. USA* **2005**, *102*, 16569–16572. [CrossRef]
7. Schubert, A. Successive h-indices. *Scientometrics* **2007**, *70*, 201–205. [CrossRef]

8. Wang, G.G.; Deb, S.; Cui, Z. Monarch butterfly optimization. *Neural Comput. Appl.* **2019**, *31*, 1995–2014. [CrossRef]
9. Wang, G.G. Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Memetic Comput.* **2018**, *10*, 151–164. [CrossRef]
10. Yi, J.H.; Xing, L.N.; Wang, G.G.; Dong, J.; Vasilakos, A.V.; Alavi, A.H.; Wang, L. Behavior of crossover operators in NSGA-III for large-scale optimization problems. *Inf. Sci.* **2020**, *509*, 470–487. [CrossRef]
11. Yi, J.H.; Deb, S.; Dong, J.; Alavi, A.H.; Wang, G.G. An improved NSGA-III algorithm with adaptive mutation operator for Big Data optimization problems. *Future Gen. Comput. Syst.* **2018**, *88*, 571–585. [CrossRef]
12. Deb, K. *Multi-Objective Optimization Using Evolutionary Algorithms*; John-Wiley: Chichester, UK, 2001.
13. Deb, K.; Agrawal, S.; Pratap, A.; Meyarivan, T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In Proceedings of the International Conference on Parallel Problem Solving From Nature, Paris, France, 18–20 December 2000; Springer: Cham, Switzerland, 2000; pp. 849–858.
14. Srinivas, N.; Deb, K. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evol. Comput.* **1994**, *2*, 221–248. [CrossRef]
15. Wang, G.G.; Tan, Y. Improving metaheuristic algorithms with information feedback models. *IEEE Trans. Cybern.* **2017**, *49*, 542–555. [CrossRef]
16. Wang, G.G.; Guo, L.; Gandomi, A.H.; Hao, G.S.; Wang, H. Chaotic krill herd algorithm. *Inf. Sci.* **2014**, *274*, 17–34. [CrossRef]
17. Sidiropoulos, A.; Gogoglou, A.; Katsaros, D.; Manolopoulos, Y. Gazing at the skyline for star scientists. *J. Informetr.* **2016**, *10*, 789–813. [CrossRef]
18. Van Leeuwen, T.; Visser, M.; Moed, H.; Nederhof, T.; Van Raan, A. The Holy Grail of science policy: Exploring and combining bibliometric tools in search of scientific excellence. *Scientometrics* **2003**, *57*, 257–280. [CrossRef]
19. Martin, B. The use of multiple indicators in the assessment of basic research. *Scientometrics* **1996**, *36*, 343–362. [CrossRef]
20. Alonso, S.; Cabrerizo, F.J.; Herrera-Viedma, E.; Herrera, F. hg-index: A new index to characterize the scientific output of researchers based on the h-and g-indices. *Scientometrics* **2010**, *82*, 391–400. [CrossRef]
21. Alonso, S.; Cabrerizo, F.J.; Herrera-Viedma, E.; Herrera, F. h-Index: A review focused in its variants, computation and standardization for different scientific fields. *J. Informetr.* **2009**, *3*, 273–289. [CrossRef]
22. Van Raan, A.F. Comparison of the Hirsch-index with standard bibliometric indicators and with peer judgment for 147 chemistry research groups. *Scientometrics* **2006**, *67*, 491–502. [CrossRef]
23. Liang, L. H-index sequence and h-index matrix: Constructions and applications. *Scientometrics* **2006**, *69*, 153–159. [CrossRef]
24. Costas, R.; Bordons, M. The h-index: Advantages, limitations and its relation with other bibliometric indicators at the micro level. *J. Informetr.* **2007**, *1*, 193–203. [CrossRef]
25. Hirsch, J.E. Does the h index have predictive power? *Proc. Natl. Acad. Sci. USA* **2007**, *104*, 19193–19198. [CrossRef]
26. Jin, B.; Liang, L.; Rousseau, R.; Egghe, L. The R-and AR-indices: Complementing the h-index. *Chin. Sci. Bull.* **2007**, *52*, 855–863. [CrossRef]
27. Bornmann, L.; Daniel, H.D. The state of h index research. *EMBO Rep.* **2009**, *10*, 2–6. [CrossRef]
28. Egghe, L. Theory and practise of the g-index. *Scientometrics* **2006**, *69*, 131–152. [CrossRef]
29. Batista, P.D.; Campiteli, M.G.; Kinouchi, O. Is it possible to compare researchers with different scientific interests? *Scientometrics* **2006**, *68*, 179–189. [CrossRef]
30. Jin, B. H-index: An evaluation indicator proposed by scientist. *Sci. Focus* **2006**, *1*, 8–9.
31. Jin, B. The AR-index: Complementing the h-index. *ISSI Newsl.* **2007**, *3*, 6.
32. Egghe, L.; Rousseau, R. An h-index weighted by citation impact. *Inf. Process. Manag.* **2008**, *44*, 770–780. [CrossRef]
33. Aguillo, I.F.; Bar-Ilan, J.; Levene, M.; Ortega, J.L. Comparing university rankings. *Scientometrics* **2010**, *85*, 243–256. [CrossRef]
34. Olcay, G.A.; Bulu, M. Is measuring the knowledge creation of universities possible? A review of university rankings. *Technol. Forecast. Soc. Chang.* **2016**, *123*, 153–160. [CrossRef]
35. Moed, H.F. A critical comparative analysis of five world university rankings. *Scientometrics* **2016**, *110*, 967–990. [CrossRef]

36. Avralev, N. Comparative Analysis of the Role of the University Ranking Positions under Conditions of Globalization in the Motivation of Prospective Students in 2011–2014. *Glob. Media J.* **2016**,*10*, 1–7.
37. Kapur, N.; Lytkin, N.; Chen, B.C.; Agarwal, D.; Perisic, I. Ranking Universities Based on Career Outcomes of Graduates. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; ACM: New York, NY, USA, 2016; pp. 137–144.
38. Aguillo, I.F. University rankings: The web ranking. *High. Learn. Res. Commun.* **2012**, *2*, 3–22. [CrossRef]
39. Çakır, M.P.; Acartürk, C.; Alaşehir, O.; Çilingir, C. A comparative analysis of global and national university ranking systems. *Scientometrics* **2015**, *103*, 813–848. [CrossRef]
40. Lukman, R.; Krajnc, D.; Glavič, P. University ranking using research, educational and environmental indicators. *J. Clean. Prod.* **2010**, *18*, 619–628. [CrossRef]
41. Baty, P. Global Rankings: Change for the better. In *The World University Rankings, Times Higher Education*; Routlege: London, UK, 2011; Volume 6.
42. Lazaridis, T. Ranking university departments using the mean h-index. *Scientometrics* **2010**, *82*, 211–216. [CrossRef]
43. Liu, N.C.; Cheng, Y. The academic ranking of world universities. *High. Educ. Europe* **2005**, *30*, 127–136. [CrossRef]
44. Dehon, C.; McCathie, A.; Verardi, V. Uncovering excellence in academic rankings: A closer look at the Shanghai ranking. *Scientometrics* **2010**, *83*, 515–524. [CrossRef]
45. The Scimago Institutions Rankings. Available online: http://www.scimagoir.com/methodology.php (accessed on 20 March 2019).
46. Aguillo, I.F.; Granadino, B.; Ortega, J.L.; Prieto, J.A. Scientific research activity and communication measured with cybermetrics indicators. *J. Am. Soc. Inf. Sci. Technol.* **2006**, *57*, 1296–1302. [CrossRef]
47. Aguillo, I.F.; Ortega, J.L.; Fernández, M. Webometric ranking of world universities: Introduction, methodology, and future developments. *High. Educ. Europe* **2008**, *33*, 233–244. [CrossRef]
48. Huang, M.H. Performance ranking of scientific papers for world universities. In Proceedings of the International Symposium: Ranking in Higher Education on the Global and National stages, Taiwan, 30–31 May 2008.
49. Radojicic, Z.; Jeremic, V. Quantity or quality: What matters more in ranking higher education institutions. *Curr. Sci.* **2012**, *103*, 158–162.
50. Jeremic, V.; Bulajic, M.; Martic, M.; Radojicic, Z. A fresh approach to evaluating the academic ranking of world universities. *Scientometrics* **2011**, *87*, 587–596. [CrossRef]
51. Ivanovic, B. *Classification Theory*; Institute for Industrial Economics: Belgrade, Serbia, 1977.
52. Leiden Ranking Website. Available online: http://www.leidenranking.com/information/indicators (accessed on 20 May 2019).
53. Van Vught, F.A.; Westerheijden, D.F. *Multidimensional Ranking*; Springer: Cham, Switzerland, 2012; pp. 11–23.
54. Van Vught, F.; Westerheijden, D.F. Multidimensional ranking. *High. Educ. Manag. Policy* **2010**, *22*, 1–26. [CrossRef]
55. Parris, T.M.; Kates, R.W. Characterizing and measuring sustainable development. *Annu. Rev. Environ. Resour.* **2003**, *28*, 559–586. [CrossRef]
56. Mayer, A.L. Strengths and weaknesses of common sustainability indices for multidimensional systems. *Environ. Int.* **2008**, *34*, 277–291. [CrossRef]
57. Afgan, N.H.; Carvalho, M.G. Sustainability assessment of hydrogen energy systems. *Int. J. Hydrogen Energy* **2004**, *29*, 1327–1342. [CrossRef]
58. Pareto, V. *Cours d'Économie Politique*; Librairie Droz: Geneva, Switzerland, 1964; Volume 1.
59. Chinchuluun, A.; Pardalos, P.M. A survey of recent developments in multiobjective optimization. *Ann. Oper. Res.* **2007**, *154*, 29–50. [CrossRef]
60. Gu, Z.M.; Wang, G.G. Improving NSGA-III algorithms with information feedback models for large-scale many-objective optimization. *Future Gen. Comput. Syst.* **2020**, *107*, 49–69. [CrossRef]
61. Talbi, E.G. *Metaheuristics: From Design to Implementation*; John Wiley & Sons: New York, NY, USA, 2009; Volume 74.
62. Kukkonen, S.; Lampinen, J. Ranking-dominance and many-objective optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, CEC IEEE, Singapore, 25–28 September 2007; pp. 3983–3990.

63. Fonseca, C.M.; Fleming, P.J. Genetic Algorithms for Multiobjective Optimization: Formulation Discussion and Generalization. In *Icga*; Citeseer: Pennsylvania, PA, USA, 1993; Volume 93, pp. 416–423.

mathematics

MDPI

Article

Hybrid Annealing Krill Herd and Quantum-Behaved Particle Swarm Optimization

Cheng-Long Wei [1] and Gai-Ge Wang [1,2,3,4,*]

[1] Department of Computer Science and Technology, Ocean University of China, Qingdao 266100, China; weichenglong@stu.ouc.edu.cn
[2] Institute of Algorithm and Big Data Analysis, Northeast Normal University, Changchun 130117, China
[3] School of Computer Science and Information Technology, Northeast Normal University, Changchun 130117, China
[4] Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China
* Correspondence: wgg@ouc.edu.cn

Received: 27 July 2020; Accepted: 17 August 2020; Published: 21 August 2020

Abstract: The particle swarm optimization algorithm (PSO) is not good at dealing with discrete optimization problems, and for the krill herd algorithm (KH), the ability of local search is relatively poor. In this paper, we optimized PSO by quantum behavior and optimized KH by simulated annealing, so a new hybrid algorithm, named the annealing krill quantum particle swarm optimization (AKQPSO) algorithm, is proposed, and is based on the annealing krill herd algorithm (AKH) and quantum particle swarm optimization algorithm (QPSO). QPSO has better performance in exploitation and AKH has better performance in exploration, so AKQPSO proposed on this basis increases the diversity of population individuals, and shows better performance in both exploitation and exploration. In addition, the quantum behavior increased the diversity of the population, and the simulated annealing strategy made the algorithm avoid falling into the local optimal value, which made the algorithm obtain better performance. The test set used in this paper is a classic 100-Digit Challenge problem, which was proposed at 2019 IEEE Congress on Evolutionary Computation (CEC 2019), and AKQPSO has achieved better performance on benchmark problems.

Keywords: swarm intelligence; simulated annealing; krill herd; particle swarm optimization; quantum

1. Introduction

With the development of modern technology, artificial intelligence is becoming more and more important in society, and more and more mature, and can be used to deal with many problems that cannot be solved by traditional methods, such as wind energy decision system (WEDS) [1] and social cognitive radio network (SCRN) [2]. There are many kinds of classification methods; the simplest classification method is divided into the traditional method and modern intelligent method [3]. The traditional optimization algorithms generally deal with structured problems. The algorithms are deterministic and have only one global optimal solution. Meanwhile, the intelligent optimization algorithms generally deal with a more general description of the problems, which is heuristic, and have multiple extreme values. Intelligent optimization algorithms require certain strategies to prevent falling into the local optimum and try to find the global optimum, such as scheduling [4–7], image [8–10], feature selection [11–13] and detection [14,15], path planning [16,17], cyber-physical social system [18,19], texture discrimination [20], factor evaluation [21], saliency detection [22], classification [23,24], object extraction [25], gesture segmentation [26], economic load dispatch [27,28], shape design [29], big data and large-scale optimization [30,31], signal processing [32], multi-objective optimization [33,34], big data optimization [30,31], unit commitment [35], vehicle routing [36], knapsack problem [37–39], fault

diagnosis [40–42], and test-sheet composition [43]. Because intelligent algorithms are not strictly dependent on the mathematical relationship, and because the problems that need to be solved are becoming more and more complex, intelligent algorithms are widely used to solve various complex optimization problems. As a result, the frequency of intelligent algorithms has exceeded that of traditional algorithms. At present, some main intelligent algorithms are widely used, such as genetic algorithm (GA) [44,45], particle swarm optimization algorithm (PSO) [46–48], krill herd (KH) [49–54], ant colony optimization algorithm (ACO) [55–57], differential evolution algorithm (DE) [58,59], adaptive island evolutionary algorithm (AIE) [60], and delayed start parallel evolutionary algorithm (DSPE) [61].

The swarm intelligence algorithms mainly come from the bionic idea, which is a set of methods summarized by referring to the laws of life and predatory behavior of animals in the biological world. A swarm intelligence algorithm is a kind of algorithm that works by studying the collective behavior of animals, and the most famous swarm intelligence algorithms are particle swarm optimization algorithm (PSO) [46] and ant colony optimization algorithm (ACO) [55]. An evolutionary algorithm is mainly a kind of method obtained by studying the process of biological genetic change, and these include genetic algorithm (GA) [44], genetic programming (GP) [62], evolutionary strategy (ES) [63,64], differential evolution (DE) [58] and other algorithms are based on genes. The krill herd (KH) [49] and quantum-behaved particle swarm optimization (QPSO) [65] algorithms mentioned in this paper belong to the swarm intelligence algorithm. The KH algorithm has the advantages of simplicity, flexibility, and high computational efficiency, and the QPSO algorithm has a relatively fast convergence speed, but when they are used to solve the 100-Digit Challenge problems, they do not do well. Owing to the poor local optimization ability of QPSO, it is easy to fall into the local optimal value, so it does not solve this problem well. In order to study an algorithm with very high accuracy, as well as to simultaneously optimize the exploitation and exploration and improve the accuracy of annealing krill quantum particle swarm optimization (AKQPSO), we studied the KH algorithm with strong exploitation and the QPSO algorithm with strong exploration. By combining their advantages, the new algorithm overcomes their original shortcomings and has a strong ability in exploration and exploitation. We combine the advantages of KH and QPSO, and optimize them, forming the annealing krill quantum particle swarm optimization (AKQPSO) algorithm. The new algorithm solved the 100-Digit Challenge better. Moreover, the study solved the problem of the poor accuracy of general algorithms, and exploitation and exploration cannot achieve the optimal at the same time.

In this paper, the 100-Digit Challenge problem is difficult to solve by traditional methods, and we can solve this problem better by using a swarm intelligence algorithm. In 2002, academician Nick Trefethen of Oxford University and the Society for Industrial and Applied Mathematics (SIAM) jointly developed the 100-Digit Challenge problem, which was proposed mainly to test high-precision computing [18]. The challenge consists of ten problems, each of which needs to be accurate to ten decimal places, with a total of 100 digits, so the problem is named the 100-Digit Challenge. However, traditional methods need a lot of computation to solve this challenge, and they cannot get good results, so we use swarm intelligence algorithm to solve these problems, and get satisfactory results.

The rest of paper is organized as follows. Most representative studies regarding the KH and QPSO algorithm are reviewed in Section 2, and these two algorithms are introduced in Section 3. The proposed algorithm is described in Section 4 and the experimental results of AKQPSO are presented in Section 5. Section 6 provides the main conclusions and highlights future work.

2. Related Work

In order to illustrate the hybrid method, we introduced KH and PSO algorithms. In the following, we will introduce the KH algorithm, PSO algorithm, and 100-Digit Challenge problem, respectively.

2.1. KH

At present, we have two kinds of optimizations for the KH algorithm [49]; one is to improve the KH algorithm itself, and the other is to improve the KH by other excellent operators or algorithms.

Wang et al. [54] changed the parameters of the KH algorithm and improved the speed of global convergence through chaos theory. Then, in order to improve the performance, they [66] used harmony search to replace the physical diffusion, which greatly improved the performance and efficiency. These algorithms belong to the first category, which optimized the KH algorithm itself. The following are the second category, which optimized the algorithm with better strategies. Abualigah et al. [50] put forward a new algorithm, which combined harmony search algorithm with he KH algorithm to generate a new probability factor and improved exploration search ability, so as to get a better solution. Another algorithm is proposed by Abualigah et al. [67], which combined the objective function with the KH algorithm, and had better performance in solving the problem of text clustering. Niu et al. [68] sped up the exploration convergence and improved the efficiency by using the opposite learning strategy, using a sinusoidal graph to change the inertia weight, and modifying the search process according to the inertia weight and acceleration coefficient.

2.2. PSO

The PSO algorithm is one of the most famous swarm intelligence algorithms [47,69,70]. It was proposed in 1995, and it has the advantages of few parameters, a simple principle, and fast convergence. This algorithm has been widely used. For example, Sun et al. [71] used the agent model to assist the PSO algorithm to solve complex optimization problems. Through the combination of exploration and exploitation, they can better solve high-dimensional problems with limited resources. Similarly, for high-dimensional problems, Tran et al. [72] changed the fixed length of feature selection in the PSO algorithm, so that the particle swarm had a shorter length. This operation reduced the search space and gave shorter particles better performance. Thus, the overall efficiency of the PSO algorithm is improved, and it is also more suitable for high-dimensional problems. For the feature selection of the PSO algorithm, they [73] also optimized other methods. They improved the algorithm by discretizing the feature selection, and proved that the single variable discretization may reduce the performance of the feature selection stage, so they proposed a new discretization method, and got better performance. Zhang et al. [74] considered that there was no PSO algorithm that can work in noisy and noiseless environment at the same time, so they proposed the dual-environment PSO algorithm. This new algorithm is based on the top-k elite particles to search, which not only guaranteed the good performance in the noise environment, but also can be re-applied to the noise-free environment, so it filled a gap in the field of the PSO algorithm.

2.3. 100-Digit Challenge

The 100-Digit Challenge problem was originally proposed by SIAM [75], which is a test method for accuracy calculation, and has been studied by many experts. Many algorithms have been applied to this challenge, and some of them are hybrid algorithms that are used to challenge this problem. Epstein et al. [76] combined the genetic algorithm (GA) with differential evolution (DE) algorithm to search the genetic space and find the latest solution. The new algorithm improved the ability to find the right answer through the fitness-based opposition and tide mutation. Brest et al. [77] used the self-adaptive differential evolution (jDE) algorithm in the DE algorithm to solve this problem, and also got better results. Different from the above algorithms, Zhang et al. [78] proposed a new DE algorithm named collective information powered DE. This new algorithm proposed a restart mechanism through collective population information to improve the robustness.

3. KH and PSO

KH and PSO are very efficient algorithms in the field of swarm intelligence, and we will respectively introduce them in the following.

3.1. KH

In the process of predation, the predator will change the distribution of krill population, which will make them move rapidly, and then cause their distribution density to decrease and the distance between the predator and the food to become more and more far, which is the initial stage of KH. In this process, the distribution of krill population is determined by the following three situations: the influence of other krill individuals, behavior of getting food, and random diffusion. The KH algorithm can be described as follows:

$$\frac{dX_i}{dt} = N_i + F_i + D_i \tag{1}$$

where N_i is the influence of other krill individuals, F_i is the behavior of getting food, and D_i is the behavior of random diffusion; $i = 1, 2, \ldots, N$, and N is the population size.

For the influence of other krill individuals, the motion $N_{i,new}$ of krill i induced by other krill is defined as follows:

$$N_{i,new} = N_{\max}\alpha_i + \omega_n N_{i,old} \tag{2}$$

where $N_{\max}$ represents the maximum induced velocity, $N_{i,old}$ represents the previously induced movement, ω_n represents the inertia weight and the value range is (0,1), and α_i indicates that the individual$_i$ is affected by the induction direction of the surrounding neighbors.

The next behavior F_i is to get food, as follows:

$$F_i = V_f\beta_i + \omega_f F_{i,old} \tag{3}$$

where V_f is the maximum foraging speed, and its value is a constant, which is 0.02 (ms^{-1}); ω_f is the inertia weight of foraging movement, and its range is (0, 1); $F_{i,old}$ is the previous foraging movement; and β_i is the foraging direction.

The individual D_i in the last behavior can be represented as follows:

$$D_i = D_{\max}(1 - \frac{I}{I_{\max}})\delta \tag{4}$$

where $D_{\max}$ represents the maximum random diffusion speed; δ represents the direction of random diffusion; and I and $I_{\max}$ represent the current number and the maximum number of iterations, respectively. From above process, we can get the krill update process of the KH algorithm as follows:

$$X_i(t + \Delta t) = X_i(t) + \Delta t\frac{dX_i}{dt} \tag{5}$$

$$\Delta t = Ct\sum_{j=1}^{NV}(UB_j - LB_j) \tag{6}$$

where Δt is the time interval related to the specific application; NV is the dimension of the decision variable; step factor Ct is a constant between 0 and 2; and UB_j and LB_j are the upper and lower bounds of corresponding variable j ($j = 1, 2, \ldots, NV$), respectively.

The process of the KH algorithm (Algorithm 1) is as follows.

Algorithm 1. KH [49]

1. Begin
2. Step 1: Initialization. Initialize the generation counter G, the population P, V_f, D_{max}, and N_{max}.
3. Step 2: Fitness calculation. Calculate fitness for each krill according to its initial position.
4. Step 3: While G < MaxGeneration **do**
    5. Sort the population according to their fitness.
    6. **for** i = 1:N (all krill) **do**
    7. Perform the following motion calculation.
        8. Motion induced by other individuals
        9. Foraging motion
        10. Physical diffusion
    11. Implement the genetic operators.
    12. Update the krill position in the search space.
    13. Calculate fitness for each krill according to its new position
14. **end for** i
15. $G = G + 1$.
16. **Step 4: end while.**
17. **End**.

3.2. PSO

In the PSO algorithm, each individual is called a "particle", and each particle will find a potential optimal solution at each iteration. Assuming that the size of the population is N, each particle i (i = 1, 2, ... , N) in the population has its own initial position X_i ($X_i = x_{i1}, x_{i2}, \ldots, x_{id}$) and initial velocity V_i ($V_i = v_{i1}, v_{i2}, \ldots, v_{id}$), and they will search for the optimal solution in D-dimensional space according to their own individual extremum p_{best} and global extremum g_{best}. Individual extremum is the current best point found by each particle in the search space, and global extremum is the current best point found by the whole particle group in the search space. During the search process, the updating formula of particle's relevant state parameters is as follows:

$$v_i^{t+1} = \eta v_i^t + c_1 * rand1() \cdot (p_{best}^t - x_i^t) + c_2 * rand2() \cdot (g_{best}^t - x_i^t) \tag{7}$$

$$x_i^{t+1} = x_i^t + v_i^t \tag{8}$$

$$\eta = \eta_{start} - (\eta_{start} - \eta_{end}) \frac{t}{T} \tag{9}$$

where η is the inertia weight that determines the specific gravity of the particle to the current velocity. If the η is large, the particle has a strong exploration ability and can span a longer distance to find the global optimal solution, but it may cause the particle to oscillate back and forth before and after the optimal solution. If the η is small, it means that particles have better ability to find the optimal solution locally, but they can easily to fall into the local optimization solution.

The flow of the standard PSO algorithm (Algorithm 2) is given below, where b_{up} and b_{lo} represent the upper and lower bounds of the problem domain, respectively, and D represents the dimension of the solution space.

Algorithm 2. PSO [46]

Begin

Step 1: Initialization.

1. **Initial position:** the initial position of each particle obeys uniform distribution, that is $X_i \sim U(b_{up}, b_{lo})$.
2. **Initialize its own optimal solution and the overall optimal solution:** the initial position is its own optimal solution $p_i = x_i$, and then calculating the corresponding value of each particle according to the defined utility function f, and find the global optimal solution g_{best}.
3. **Initial speed:** the speed also obeys the uniform distribution.

Step 2: Update.

According to Equations (7) and (8), **the velocity and position of particles are updated,** and the current fitness of particles is calculated according to the utility function f of the problem. If it is better than its own historical optimal solution, it will update its own historical optimal solution p_{best}, otherwise it will not update. If the particle's own optimal solution p_{best} is better than the global optimal solution g, then the global optimal solution g is updated, otherwise it is not updated.

Step 3: Determine whether to terminate:

Determine whether the best solution meets the termination conditions, if yes, stop.

Otherwise, return to **Step 2**.

End.

As shown in Equations (7) and (8), during each iteration, the particles update the direction and speed of the next flight based on their own and group experience. The main characteristics of the PSO algorithm are as follows:

1. Particles have memory. Each iteration of particles will transfer the optimal solution of the population to each other, and update the database of all particles. If the particles deviate, the direction and velocity can be corrected by self-cognition and group-cognition.
2. PSO algorithm has fewer parameters, so it is easy to adjust, and the structure is simple, so it is easy to implement.
3. The operation is simple, and it only searches for the optimal solution in the solution space according to the flight of particles.

4. AKQPSO

The AKQPSO algorithm is a mixed metaheuristic algorithm, which combines the advantages of annealing krill herd (AKH) and QPSO [79]. AKH solves the disadvantage that KH cannot escape from the local optimal solution. At the same time, the efficiency of KH algorithm is improved by the simulated annealing strategy. Quantum behaved PSO (QPSO) is a new algorithm proposed by Sun et al. [79] in 2004. By introducing the quantum behavior and combining it with the idea of simulated annealing, the search ability of AKQPSO is greatly improved, and the new algorithm has better performance. The principle of QPSO is shown below.

Quantum computing is a new computing mode, which follows the laws of quantum mechanics and regulates the quantum information unit. In quantum space, when the aggregation state property is satisfied, particles can search in the whole feasible solution space, thus greatly improving the exploration search ability of QPSO. According to the analysis theory of particle convergence trajectory, if every particle can converge to its local attraction point $P_i = (p_{i1}, p_{i2} \ldots, p_{id})$, then the algorithm has the possibility of convergence. The particle position update expression of the standard QPSO algorithm is as follows:

$$x_{ij}^{t+1} = p_{ij}^{t} \pm \alpha \cdot \left| C_{ij}^{t} - x_{ij}^{t} \right| \cdot \ln[1/u_{ij}^{t}], u_{ij}^{t} \sim U(0,1) \tag{10}$$

where α is the only parameter that needs to be adjusted in the algorithm, called the compression-expansion factor, which is used to control the convergence rate of particles. During the iterative process, the calculation method of individual and global optimal position is the same

as that of the PSO algorithm, and the biggest difference is that the QPSO algorithm removes the speed information.

First of all, we initialize the whole population, and all the individuals in the population are randomly generated. After initialization, we divide the population into two subpopulations according to the ratio of 3:7, which is discussed in Section 5.2. The population with the proportion of 3/10 is optimized by the improved KH algorithm, which is called subpopulation-AKH, and the population with the proportion of 7/10 is optimized by the QPSO algorithm, which is called subpopulation-QPSO. The two subpopulations will be re-integrated into a population after iterative optimization. If the optimization result of the new population meets the output conditions, then the result can be output. However, if it does not meet the output conditions, then it can be re-divided according to the proportion of 3:7, and repeat the above process until the results meet the termination conditions. In the process of decentralized and re-fusion of this population, the location information of individuals is shared, so it will greatly improve the efficiency of fusion search and get the best results faster. In the process of the AKQPSO algorithm, we also optimized the KH algorithm by the idea of simulated annealing, named annealing KH (AKH), so we used QPSO and simulated annealing strategy to guarantee the algorithm to escape from the local optimal value. The framework of AKQPSO is shown in Figure 1 and the process of the AKQPSO algorithm (Algorithm 3) is as follows.

Algorithm 3 AKQPSO

Initialization:
N random individuals were generated.
Set initialization parameters of AKH and QPSO.

Evaluation:
Evaluate all individuals based on location.

Partition:
The whole population was divided into subpopulation-AKH and subpopulation-QPSO.

AKH process:
Subpopulation-AKH individuals were optimized by AKH.
Update through these three actions of the influence of other krill individuals, behavior of getting food and random diffusion.
The simulated annealing strategy is used to deal with the above behaviors.
Update the individual position according to the above behavior.

QPSO process:
Subpopulation-QPSO individuals were optimized by QPSO.
Update the particle's local best point P_i and global best point P_{best}.
Update the position x_{ij}^{t+1} by Equation (10).

Combination:
The population optimized by AKH and QPSO was reconstituted into a new population.

Finding the best solution:
The fitness of all individuals was calculated, and the best solution was found in the newly-combined population.

Determine whether to terminate:
Determine whether the best solution meets the termination conditions, if yes, stop.
Otherwise, return to step **Evaluation**.

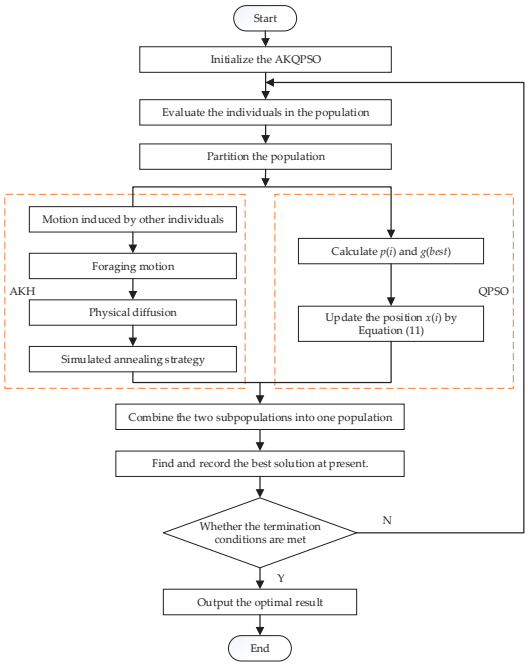


Figure 1. Framework of annealing krill quantum particle swarm optimization (AKQPSO).

5. Simulation Results

The 100-Digit Challenge problem comes from CEC 2019 and can be found through the website http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2019. This challenge includes 10 problems, which are 10 functions in our data. The goal is to calculate the accuracy of the function to 10 digits without a time limit. The proposed algorithm was written in MATLAB R2016a and was run in the following conditions: Intel(R) Core(TM) i5-8100 CPU 3.60 GHz, which possesses 8G of RAM on Windows 10. Table 1 shows the basic parameters of the 100-Digit Challenge. The 10 test problems are the definition of minimization and some other definitions are as follows:

$$\text{Min} \quad f(x), x = [x_1, x_2, \ldots, x_D]^T \tag{11}$$

where D is the dimension, the global optimal value of the shift is randomly distributed in (−80, 80), and all testing problems are scalable.

Table 1. The basic parameters of the 100-Digit Challenge.

No.	Functions	$F_i^* = F_i(x^*)$	D	Search Range
1	Storn's Chebyshev Polynomial Fitting Problem	1	9	(−8192–8192)
2	Inverse Hilbert Matrix Problem	1	16	(−16,384–16,384)
3	Lennard–Jones Minimum Energy Cluster	1	18	(−4–4)
4	Rastrigin's Function	1	10	(−100–100)
5	Griewangk's Function	1	10	(−100–100)
6	Weierstrass Function	1	10	(−100–100)
7	Modified Schwefel's Function	1	10	(−100–100)
8	Expanded Schaffer's F6 Function	1	10	(−100–100)
9	Happy Cat Function	1	10	(−100–100)
10	Ackley Function	1	10	(−100–100)

5.1. The Comparison of AKQPSO and Other Algorithms

In this paper, we compared the AKQPSO algorithm with the following eight state-of-the-art algorithms.

1. By improving krill migration operator, biogeography-based KH (BBKH) [51] was proposed.
2. By adding a new hybrid differential evolution operator, the efficiency of the updating process is improved, and differential evolution KH (DEKH) [80] was proposed.
3. By quantum behavior to optimize KH, quantum-behaved KH (QKH) [65] was proposed.
4. By adding the stud selection and crossover operator, the efficiency was improved and stud krill herd (SKH) [81] was proposed.
5. By adding chaos map to optimize cuckoo search (CS), the chaotic CS (CCS) [82] was proposed.
6. By adding variable neighborhood (VN) search to bat algorithm (BA), VNBA [83] was proposed.
7. By discussing physical biogeography and its mathematics, biogeography-based optimization (BBO) [84] was proposed.
8. Genetic algorithm (GA) [45] is basic algorithm of evolutionary computing.

Table 2 is the common parameter setting of these algorithms. We set the population size (N) to 50; after 7500 iterations (*max_gen*) and 100 independent runs (*max_run*), all algorithm experiments are carried out under the condition that the basic parameters are consistent.

Table 2. Parameter settings.

Parameters	***N***	***max_gen***	***max_run***
Value	50	7500	100

Table 3 shows the optimization results of AKQPSO and eight other algorithms for ten of the 100-Digits Challenge, **with the best value in bold**. From the experimental results, we can see that the optimal values of the ten problems are all calculated by the AKQPSO algorithm. AKQPSO is the best algorithm among these algorithms, and the result is the best in every problem, but QKH is not the second in every problem, so we can know that our algorithm improvement is very effective. Through Figure 2, we can clearly conclude that the performance of AKQPSO is the best in general, followed by QKH, and the performance of GA is the worst. In general, we rank the algorithms as follows: AKQPSO > QKH > CCS > BBKH > DEKH > SKH > VNBA > GA.

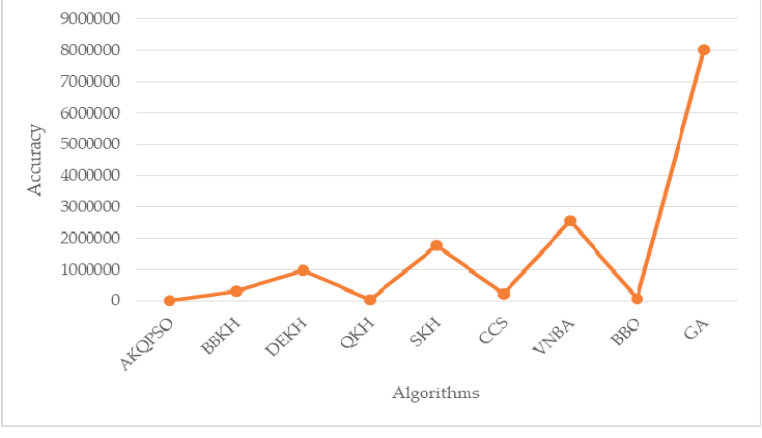


Figure 2. Accuracy of AKQPSO and other algorithms on all problems.

Table 3. The minimum value of annealing krill quantum particle swarm optimization (AKQPSO) and other algorithms in the 100-Digit Challenge. BBKH, biogeography-based krill herd; DEKH, differential evolution KH; QKH, quantum-behaved KH; SKH, stud KH; CCS, chaotic cuckoo search; VNBA, variable neighborhood bat algorithm; BBO, biogeography-based optimization; GA, genetic algorithm. (The table results retain ten decimal places.).

Algorithm	Problem 1	Problem 2	Problem 3
AKQPSO (Algorithm 3)	**10,893.3861385383**	**215.6832766302**	**1.3068652046**
BBKH	291,803.2852933580	613.7135086878	5.9027057769
DEKH	975,339.7308122220	708.0601338156	6.5872946609
QKH	47,583.9014728552	404.4416682185	2.1853645745
SKH	1,773,168.4809322700	402.7243166556	8.0692567398
CCS	211,411.5538712060	358.9175375272	12.4818284785
VNBA	2,550,944.5550316900	816.8071391350	4.6163144446
BBO	72,287.9373622652	311.8124377894	2.9209583205
GA	8,009,879.0206872900	2209.1904660292	9.7547945042
Algorithm	**Problem 4**	**Problem 5**	**Problem 6**
AKQPSO (Algorithm 3)	**4.0398322695**	**1.0434696313**	**1.1185826442**
BBKH	17.7900443880	1.8852367864	3.9551854227
DEKH	29.4088567232	1.9507021960	3.1953451545
QKH	7.1133424631	2.0695362381	3.1213610789
SKH	50.9548872356	1.9350825919	10.2543233489
CCS	61.4604530135	3.4583111771	6.6010211976
VNBA	24.7278199397	3.0251295691	5.7134938818
BBO	17.4206259198	1.0971964590	2.5154932374
GA	53.9444360765	2.7126573212	6.2420012291
Algorithm	**Problem 7**	**Problem 8**	**Problem 9**
AKQPSO (Algorithm 3)	**121.8932375270**	**2.3131324015**	**1.0715438957**
BBKH	779.4066229661	4.3846591727	1.3554069744
DEKH	1060.0387128932	4.4509969471	1.3614909209
QKH	195.0770363640	2.5711387868	1.2989726797
SKH	1348.8213407547	4.8795287164	1.5980935412
CCS	2247.4406602539	5.4002626350	1.6453371395
VNBA	644.6334455535	4.1477932707	1.3950038682
BBO	950.8018632939	3.5941524656	1.2098037868
GA	1276.4565372145	4.2393160129	1.3580083289
Algorithm	**Problem 10**	-	-
AKQPSO (Algorithm 3)	**21.0237707978**	-	-
BBKH	21.6420368754	-	-
DEKH	21.6383024320	-	-
QKH	21.1582649016	-	-
SKH	21.6017549654	-	-
CCS	21.9603961872	-	-
VNBA	21.0799825759	-	-
BBO	21.9988888954	-	-
GA	21.4392031521	-	-

Figures 3–12 show the results of the nine algorithms on each problem, and the experimental results show that AKQPSO algorithm has obvious advantages over other algorithms. From these column charts, we can see that the fluctuation of problem 1 is the largest. In problem 1, the results of the GA algorithm with the worst performance are 735 times different from those of AKQPSO algorithm with the best performance, which can be said to be very different. Because the 100-Digits Challenge problem

is used to test the computational accuracy, we can know that the computational accuracy of AKQPSO algorithm is the highest among the nine algorithms. In Figure 3, because the results of AKQPSO, QKH, and BBO are so different from those of GA, the columns of these algorithms are almost invisible, so they are not obvious in the same chart. However, in fact, the improvement of AKQPSO compared with QKH and BBO is significant.

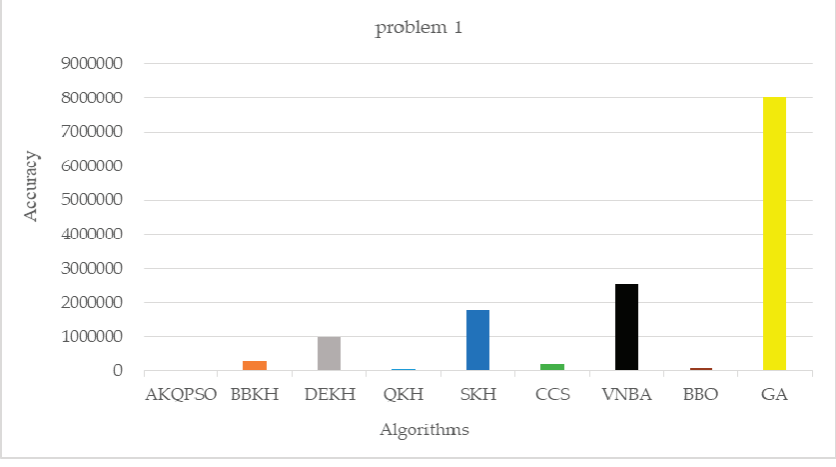


Figure 3. Accuracy of AKQPSO and other algorithms on problem 1: Storn's Chebyshev polynomial fitting problem. BBKH, biogeography-based krill herd; DEKH, differential evolution KH; QKH, quantum-behaved KH; SKH, stud KH; CCS, chaotic cuckoo search; VNBA, variable neighborhood bat algorithm; BBO, biogeography-based optimization; GA, genetic algorithm.

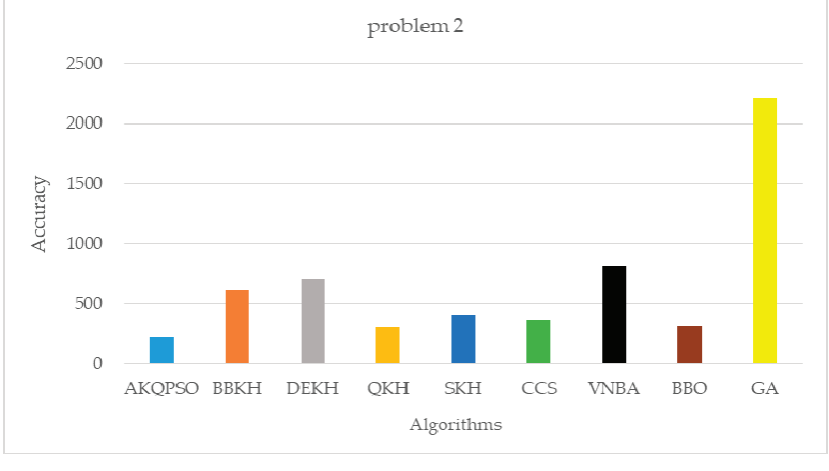


Figure 4. Accuracy of AKQPSO and other algorithms on problem 2: inverse Hilbert matrix problem.

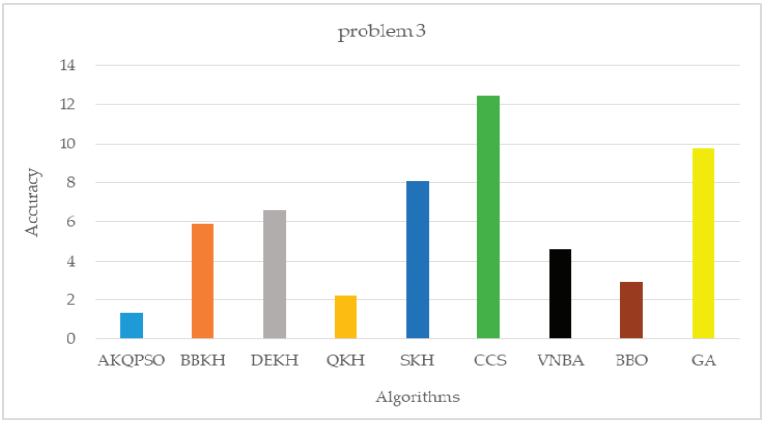


Figure 5. Accuracy of AKQPSO and other algorithms on problem 3: Lennard–Jones minimum energy cluster.

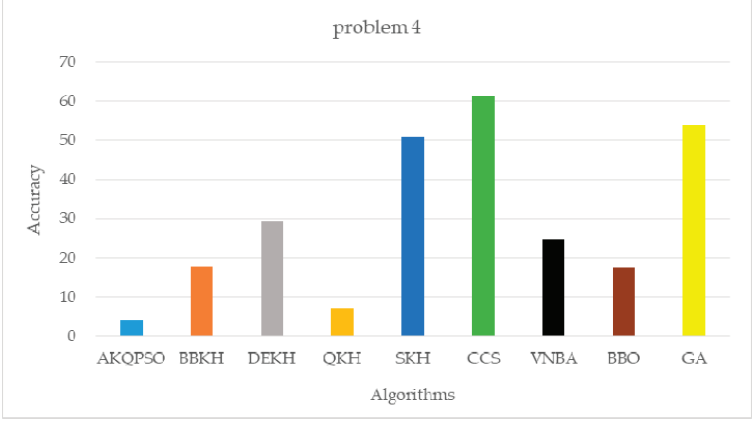


Figure 6. Accuracy of AKQPSO and other algorithms on problem 4: Rastrigin's function.

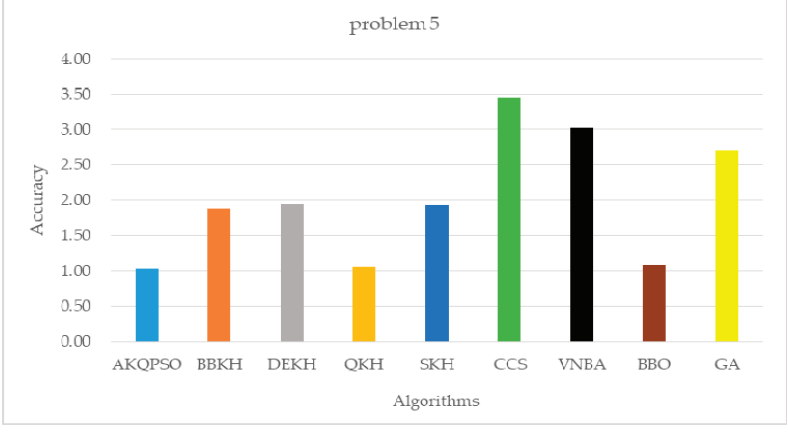


Figure 7. Accuracy of AKQPSO and other algorithms on problem 5: Griewangk's function.

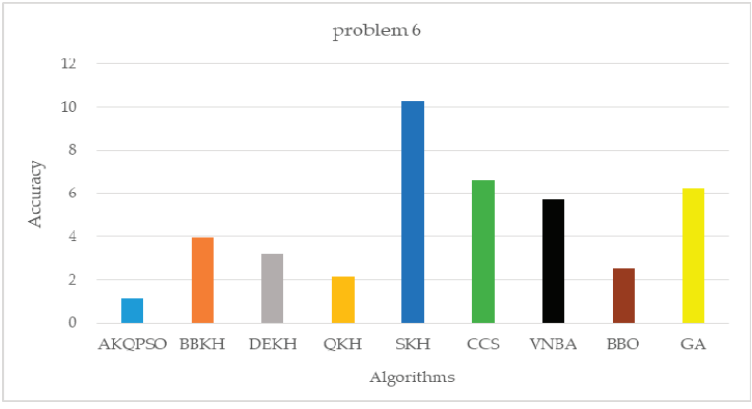


Figure 8. Accuracy of AKQPSO and other algorithms on problem 6: Weierstrass function.

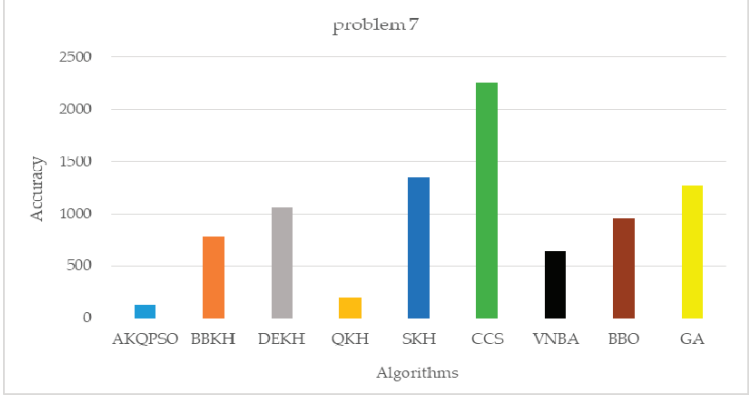


Figure 9. Accuracy of AKQPSO and other algorithms on problem 7: modified Schwefel's function.

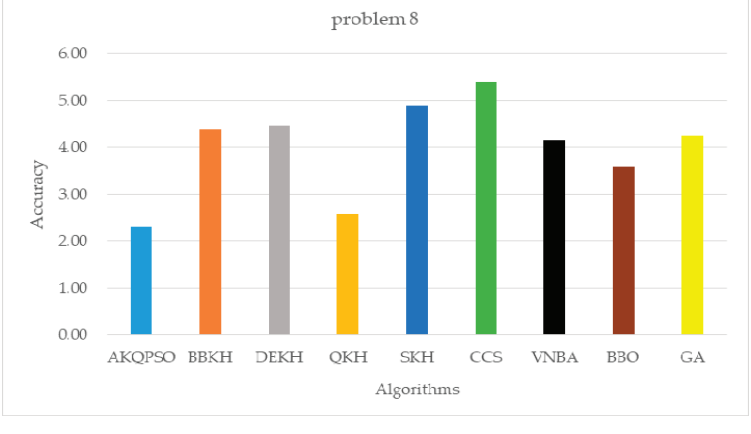


Figure 10. Accuracy of AKQPSO and other algorithms on problem 8: expanded Schaffer's F6 function.

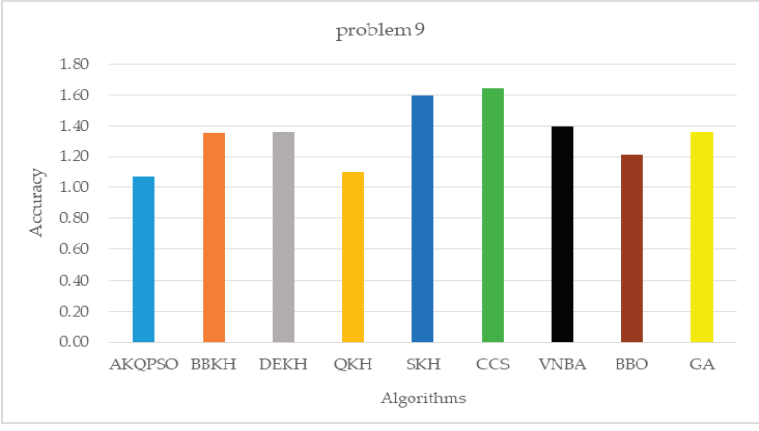


Figure 11. Accuracy of AKQPSO and other algorithms on problem 9: happy cat function.

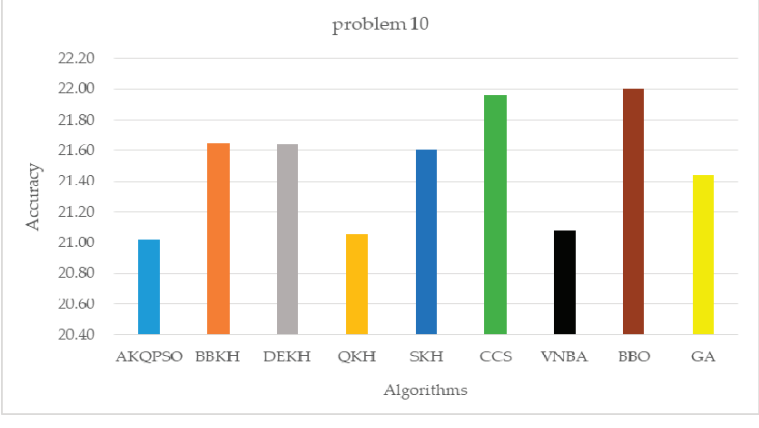


Figure 12. Accuracy of AKQPSO and other algorithms on problem 10: Ackley function.

Among these ten problems, we divide the problems into four groups according to the ability of these problems to test the computational accuracy of AKQPSO. Some groups have a strong ability to test, so it is more obvious which algorithm is better. The first group: problem 1. According to the computational accuracy, the results displayed in this group are quite different. The second group: problems 2, 3, and 7. This group has higher requirements for the accuracy, so it is difficult to calculate the best value of the function. The third group: problems 4, 6, 8, and 10. This group has a slightly lower requirement for the computational accuracy, so the results of various algorithms are very close, but it is difficult to achieve the best results. The fourth group: problems 5 and 9. This group has the lowest requirements for the computational accuracy, so many algorithms can be very close to the optimal result 1.0000000000.

Figures 13–16 show the advantage of AKQPSO over other algorithms in each group of the 100-Digits Challenge problem. Although the accuracy of GA is the worst when ranking from the whole, and in the first of the four groups, it is also the GA. In the other three groups, however, the algorithm with the worst accuracy is CCS algorithm, so in terms of grouping comparison, CCS algorithm is the worst. In addition, the accuracy of AKQPSO algorithm is still the best among the four groups, which has not changed. From the above analysis, AKQPSO is the best in all aspects, and the computational accuracy of this algorithm is also the highest.

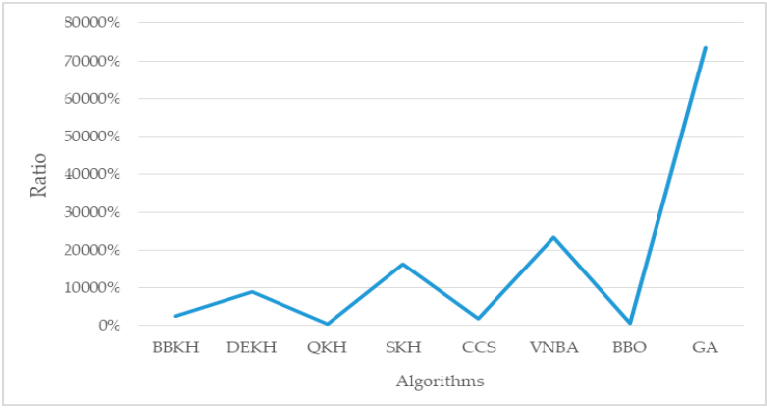


Figure 13. Ratio of AKQPSO over other algorithms in the first group.

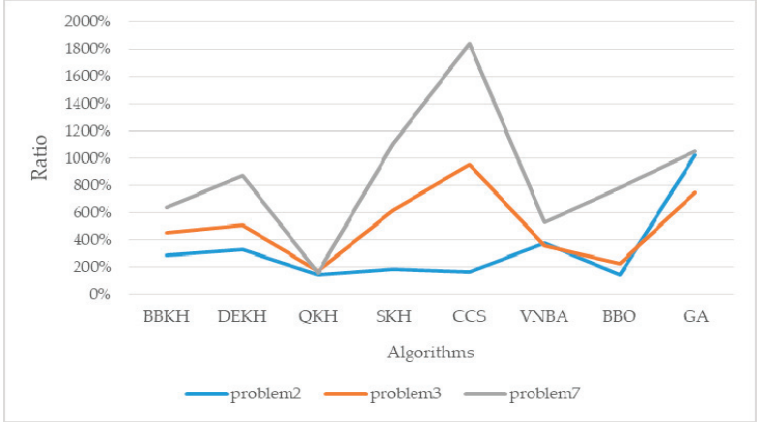


Figure 14. Ratio of AKQPSO over other algorithms in the second group.

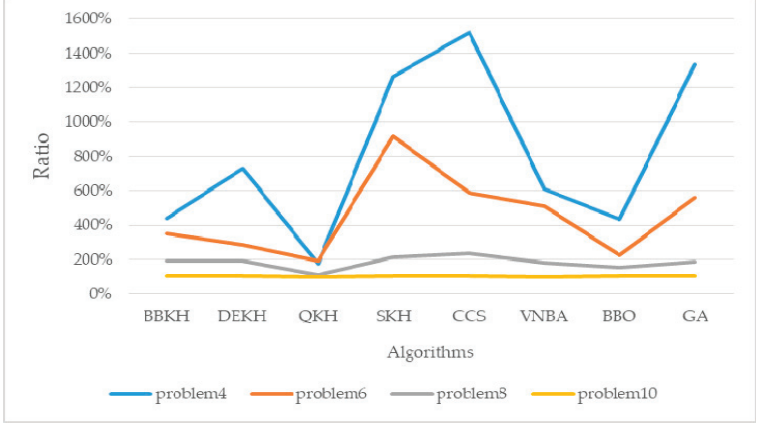


Figure 15. Ratio of AKQPSO over other algorithms in the third group.

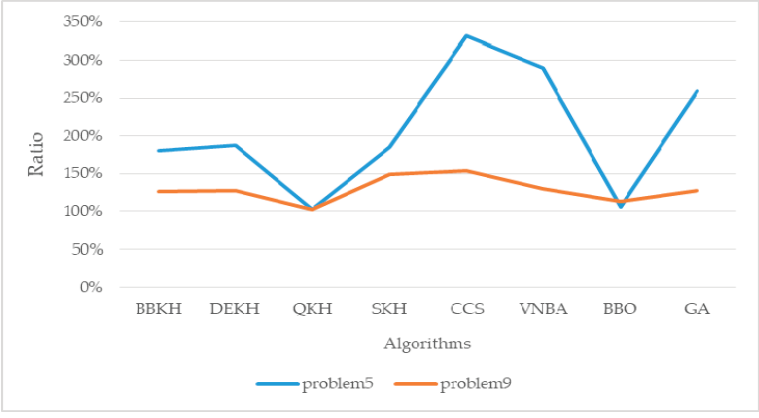


Figure 16. Ratio of AKQPSO over other algorithms in the fourth group.

5.2. Evaluation Parameter λ

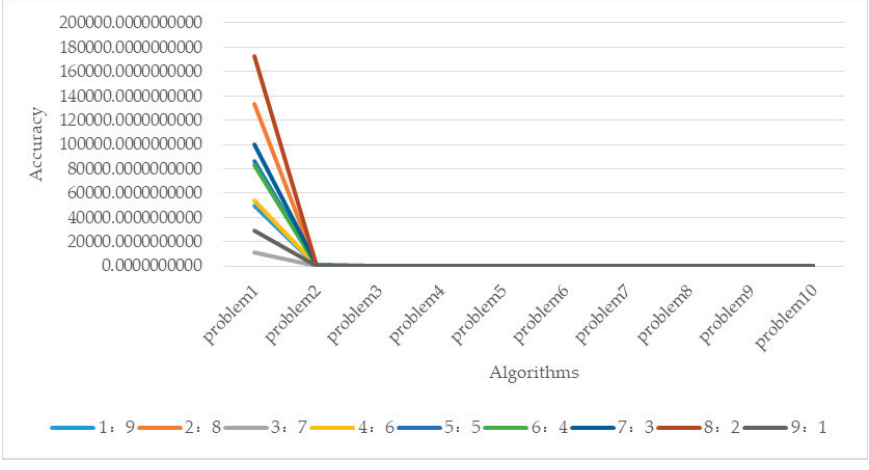
In Section 4, after the initialization of the population, AKQPSO algorithm divided the population into two subpopulations: AKH and QPSO. We set the parameter λ, which represents the proportion of the two subpopulations. In the experiment, λ = 0.1, 0.2, . . . , 0.9. For example, if λ = 0.1, this means that the ratio of subpopulation-AKH/subpopulation-QPSO is 1:9, and other numbers mean the same thing. After a lot of experiments, the optimal parameter value is λ = 0.3. All the above experiments are based on the results obtained using λ = 0.3, and the following part will introduce the experiments on λ = 0.3. The parameters of this part of experiments are still the same as Table 2 to ensure that all experiments are tested under the same conditions.

Table 4 shows the experimental results of 10 problems that are calculated by different λ in the AKQPSO algorithm in the 100-Digit Challenge. **Bold font indicates the best value**. From Table 4, we can see that, in the case of λ = 0.3, eight of the ten problems can get the best value. In the remaining two problems, even if λ = 0.3 does not reach the best value, its result is the second among the nine values, and it is very close to the best value. Therefore, when λ = 0.3, the performance of AKQPSO can reach the best.

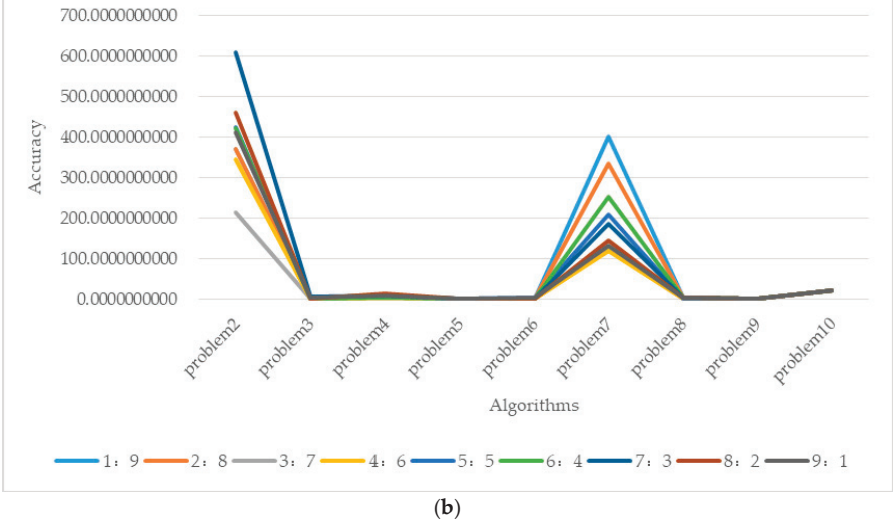
Table 4. The different subpopulations in the AKQPSO algorithm.

λ	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5
0.1	49,649.1118923043	371.5625700159	1.4091799599	6.9697543426	1.1082189445
0.2	133,608.1106469210	370.0473039979	1.4091358439	**3.9848771713**	1.0983396055
0.3	**10,893.3861385383**	**215.6832766302**	**1.3068652046**	4.0398322695	**1.0434696313**
0.4	53,539.2035950009	346.1060025716	1.4091347288	4.9798362284	1.0588561989
0.5	86,346.3589566716	424.4795442904	1.4091497973	5.9747952855	1.0564120753
0.6	82,979.4098834243	422.2483723507	1.4094790359	5.0457481023	1.0861547616
0.7	99,948.7393233432	609.3844384539	7.7057897580	7.9647083618	1.0885926731
0.8	172,794.7224327620	462.1482898652	1.4091546130	13.9344627044	1.0689273036
0.9	29,013.1541012323	412.4776865757	3.9404220234	8.9597299262	1.0642761100
λ	**Problem 6**	**Problem 7**	**Problem 8**	**Problem 9**	**Problem 10**
0.1	1.4704093392	401.1855067018	3.0118025092	1.0958187632	21.1048204811
0.2	1.2289194470	336.4312675492	2.7987816724	1.0968167661	21.0708288643
0.3	**1.1185826442**	121.8932375270	**2.3131324015**	**1.0715438957**	**21.0237707978**
0.4	2.6943320338	**119.5337169812**	2.3299026264	1.1010758378	21.0389405345
0.5	2.9771884111	209.5578114314	2.5217968033	1.1257275250	21.0587530353
0.6	2.5706601162	253.1211550958	3.5227629532	1.1353127271	21.0976573470
0.7	1.4737157145	187.5895821815	3.0884949525	1.1344605652	21.1144506567
0.8	1.5487741806	144.3502633617	3.1612574448	1.1553497252	21.0835644844
0.9	4.0000000000	131.2707035329	4.0209018951	1.1615503990	21.0768903380

As can be seen from Figure 17a, the fluctuation range of the result of problem 1 is the largest. As problems 1–10 are in the same chart, and the fluctuation of problem 1 is much larger than that of other problems, the fluctuation range of problems 2–10 is not obvious in Figure 17a. In order to compare problems 2–10 more clearly, we used Figure 17b to show the fluctuation of other problems 2–10. Through Figure 17b, we can see that the fluctuation of problems 2 and 7 are also obvious. Therefore, among these ten problems, problems 1, 2, and 7 are the three problems with the largest fluctuation. Among these ten problems, the fluctuation of problem 7 is different from that of the other nine problems. With the increasing proportion of AKH and QPSO, the result of problem 7 tends to be better. For other problems, however, with the increasing proportion of AKH and QPSO, their results tend to be worse, which is the opposite of problem 7.



(**a**)



(**b**)

Figure 17. The different subpopulations in different problems. (**a**) The different subpopulations in problems 1–10 and (**b**) the different subpopulations in problems 2–10.

From the results of Sections 5.1 and 5.2, problems 1, 2, and 7 belong to the first and the second of the four groups, respectively, and they are also the two groups with the highest requirements for the computational accuracy of AKQPSO. Therefore, the adjustment of subpopulation has a very direct effect on the accuracy of AKQPSO.

5.3. Complexity Analysis of AKQPSO

The main computing overhead of AKQPSO associated with **Step "Partition"** of the AKQPSO algorithm in Section 4. The following is a detailed analysis of the single computational complexity of **Step "Partition"** and other step in AKQPSO. N is the number of individuals in the population.

1. **Step "Partition":** This step accounts for the main computational overhead, so we focus on this step.
    - **Step "AKH process":** The computational complexity of this step mainly includes "Subpopulation-AKH individuals were optimized by AKH", "Update through these three actions of the influence of other krill individuals, behavior of getting food and random diffusion", "The simulated annealing strategy is used to deal with the above behaviors", and "Update the individual position according to the above behavior", and their complexities are O(N), O(N^2), O(N), and O(N), respectively.
    - **Step "QPSO process":** The computational complexity of this step mainly includes "Subpopulation-QPSO individuals were optimized by QPSO", "Update the particle's local best point P_i and global best point P_{best}", and "Update the position x_{ij}^{t+1} by Equation (10)", and their complexities are all O(N).
2. Other step.
    - The computational complexity of **Step "Initialization"**, **Step "Evaluation"**, **Step "Combination"**, and **Step "Finding the best solution"** are all O(N).

Therefore, in one generation AKQPSO, the worst-case complexity is O(N^2).

6. Conclusions

In this paper, aiming at the disadvantage of poor local search ability of KH algorithm, we optimized it by simulated annealing strategy and QPSO algorithm, and proposed a new algorithm: AKQPSO. This algorithm was compared with eight other excellent algorithms, and the computational accuracy of AKQPSO was tested by the 100-Digit Challenge problem. As predicted before the experiment, the computational accuracy of AKQPSO algorithm is the highest among these algorithms. Moreover, we also adjusted the parameters of AKQPSO through experiments to further improve the computational accuracy. By calculating the accuracy of the 100-Digit Challenge problem, our experiment provided a new method to study the accuracy of the algorithm, and the paper provided a very high accuracy algorithm. However, the research of this paper still has some limitations. The accuracy of the algorithm in 100-Digit Challenge problem is very high, but it is unclear whether AKQPSO still has high accuracy in general problems.

In the future, we can focus on other aspects. Firstly, besides the simulated annealing strategy and QPSO algorithm, we could look for other metaheuristic algorithms [85] that can be used to improve the search ability of KH and carry out in-depth research such as bat algorithm (BA) [86,87], biogeography-based optimization (BBO) [84,88], cuckoo search (CS) [82,89–92], earthworm optimization algorithm (EWA) [93], elephant herding optimization (EHO) [94,95], moth search (MS) algorithm [96], firefly algorithm (FA) [97], artificial bee colony (ABC) [98–100], harmony search (HS) [101], monarch butterfly optimization (MBO) [102,103], and genetic programming (GP) [104], as well as more recent research. Secondly, for the parameters in the algorithm, besides the proportion of subpopulation, we can also study the influence of other parameters on the computational accuracy of AKQPSO.

Thirdly, now we just use two kinds of algorithms to make them cooperate and optimize in a relatively simple way, and we can also study how to make them cooperate more efficiently through other methods. Finally, although the 100-Digit Challenge is a classical problem, it is still less used for testing computational accuracy of algorithms, and there is still a large development space in this area.

Author Contributions: Conceptualization, C.-L.W. and G.-G.W.; methodology, C.-L.W.; software, G.-G.W.; validation, C.-L.W.; formal analysis, G.-G.W.; investigation, C.-L.W.; resources, G.-G.W.; data curation, C.-L.W.; writing—original draft preparation, C.-L.W.; writing—review and editing, C.-L.W.; visualization, C.-L.W.; supervision, G.-G.W.; project administration, G.-G.W.; funding acquisition, G.-G.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China, grant number U1706218, 41576011, 41706010 and 61503165.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhao, X.; Wang, C.; Su, J.; Wang, J. Research and application based on the swarm intelligence algorithm and artificial intelligence for wind farm decision system. *Renew. Energy* **2019**, *134*, 681–697. [CrossRef]
2. Anandakumar, H.; Umamaheswari, K. A bio-inspired swarm intelligence technique for social aware cognitive radio handovers. *Comput. Electr. Eng.* **2018**, *71*, 925–937. [CrossRef]
3. Shang, K.; Ishibuchi, H. A New Hypervolume-based Evolutionary Algorithm for Many-objective Optimization. *IEEE Trans. Evol. Comput.* **2020**, 1. [CrossRef]
4. Sang, H.-Y.; Pan, Q.-K.; Duan, P.-Y.; Li, J.-Q. An effective discrete invasive weed optimization algorithm for lot-streaming flowshop scheduling problems. *J. Intell. Manuf.* **2015**, *29*, 1337–1349. [CrossRef]
5. Sang, H.-Y.; Pan, Q.-K.; Li, J.-Q.; Wang, P.; Han, Y.-Y.; Gao, K.-Z.; Duan, P. Effective invasive weed optimization algorithms for distributed assembly permutation flowshop problem with total flowtime criterion. *Swarm Evol. Comput.* **2019**, *44*, 64–73. [CrossRef]
6. Pan, Q.-K.; Sang, H.-Y.; Duan, J.-H.; Gao, L. An improved fruit fly optimization algorithm for continuous function optimization problems. *Knowl. Based Syst.* **2014**, *62*, 69–83. [CrossRef]
7. Gao, D.; Wang, G.-G.; Pedrycz, W. Solving Fuzzy Job-shop Scheduling Problem Using DE Algorithm Improved by a Selection Mechanism. *IEEE Trans. Fuzzy Syst.* **2020**, 1. [CrossRef]
8. Li, M.; Xiao, D.; Zhang, Y.; Nan, H. Reversible data hiding in encrypted images using cross division and additive homomorphism. *Signal Process. Image Commun.* **2015**, *39*, 234–248. [CrossRef]
9. Li, M.; Guo, Y.; Huang, J.; Li, Y. Cryptanalysis of a chaotic image encryption scheme based on permutation-diffusion structure. *Signal Process. Image Commun.* **2018**, *62*, 164–172. [CrossRef]
10. Fan, H.; Li, M.; Liu, N.; Zhang, E. Cryptanalysis of a colour image encryption using chaotic APFM nonlinear adaptive filter. *Signal Process.* **2018**, *143*, 28–41. [CrossRef]
11. Zhang, Y.; Gong, D.-W.; Hu, Y.; Zhang, W. Feature selection algorithm based on bare bones particle swarm optimization. *Neurocomputing* **2015**, *148*, 150–157. [CrossRef]
12. Zhang, Y.; Song, X.-F.; Gong, D.-W. A return-cost-based binary firefly algorithm for feature selection. *Inf. Sci.* **2017**, *418*, 561–574. [CrossRef]
13. Mao, W.; He, J.; Tang, J.; Li, Y. Predicting remaining useful life of rolling bearings based on deep feature representation and long short-term memory neural network. *Adv. Mech. Eng.* **2018**, *10*, 10. [CrossRef]
14. Jian, M.; Lam, K.M.; Dong, J. Facial-feature detection and localization based on a hierarchical scheme. *Inf. Sci.* **2014**, *262*, 1–14. [CrossRef]
15. Fan, L.; Xu, S.; Liu, D.; Ru, Y. Semi-Supervised Community Detection Based on Distance Dynamics. *IEEE Access* **2018**, *6*, 37261–37271. [CrossRef]
16. Wang, G.; Chu, H.E.; Mirjalili, S. Three-dimensional path planning for UCAV using an improved bat algorithm. *Aerosp. Sci. Technol.* **2016**, *49*, 231–238. [CrossRef]
17. Wang, G.; Guo, L.; Duan, H.; Liu, L.; Wang, H.; Shao, M. Path Planning for Uninhabited Combat Aerial Vehicle Using Hybrid Meta-Heuristic DE/BBO Algorithm. *Adv. Sci. Eng. Med.* **2012**, *4*, 550–564. [CrossRef]
18. Wang, G.; Cai, X.; Cui, Z.; Min, G.; Chen, J. High Performance Computing for Cyber Physical Social Systems by Using Evolutionary Multi-Objective Optimization Algorithm. *IEEE Trans. Emerg. Top. Comput.* **2017**, *8*, 1. [CrossRef]

19. Cui, Z.; Sun, B.; Wang, G.; Xue, Y.; Chen, J. A novel oriented cuckoo search algorithm to improve DV-Hop performance for cyber-physical systems. *J. Parallel Distrib. Comput.* **2017**, *103*, 42–52. [CrossRef]
20. Jian, M.; Lam, K.M.; Dong, J. Illumination-insensitive texture discrimination based on illumination compensation and enhancement. *Inf. Sci.* **2014**, *269*, 60–72. [CrossRef]
21. Wang, G.-G.; Guo, L.; Duan, H.; Liu, L.; Wang, H. The model and algorithm for the target threat assessment based on elman_adaboost strong predictor. *Acta Electron. Sin.* **2012**, *40*, 901–906.
22. Jian, M.; Lam, K.M.; Dong, J.; Shen, L. Visual-Patch-Attention-Aware Saliency Detection. *IEEE Trans. Cybern.* **2014**, *45*, 1575–1586. [CrossRef] [PubMed]
23. Wang, G.; Lu, M.; Dong, Y.-Q.; Zhao, X.-J. Self-adaptive extreme learning machine. *Neural Comput. Appl.* **2015**, *27*, 291–303. [CrossRef]
24. Mao, W.; Zheng, Y.; Mu, X.; Zhao, J. Uncertainty evaluation and model selection of extreme learning machine based on Riemannian metric. *Neural Comput. Appl.* **2013**, *24*, 1613–1625. [CrossRef]
25. Liu, G.; Zou, J. Level set evolution with sparsity constraint for object extraction. *IET Image Process.* **2018**, *12*, 1413–1422. [CrossRef]
26. Liu, K.; Gong, D.; Meng, F.; Chen, H.; Wang, G. Gesture segmentation based on a two-phase estimation of distribution algorithm. *Inf. Sci.* **2017**, 88–105. [CrossRef]
27. Parouha, R.P.; Das, K.N. Economic load dispatch using memory based differential evolution. *Int. J. Bio-Inspired Comput.* **2018**, *11*, 159–170. [CrossRef]
28. Rizk-Allah, R.M.; El-Sehiemy, R.A.; Wang, G. A novel parallel hurricane optimization algorithm for secure emission/economic load dispatch solution. *Appl. Soft Comput.* **2018**, *63*, 206–222. [CrossRef]
29. Rizk-Allah, R.M.; El-Sehiemy, R.A.; Deb, S.; Wang, G. A novel fruit fly framework for multi-objective shape design of tubular linear synchronous motor. *J. Supercomput.* **2017**, *73*, 1235–1256. [CrossRef]
30. Yi, J.-H.; Xing, L.; Wang, G.; Dong, J.; Vasilakos, A.V.; Alavi, A.H.; Wang, L. Behavior of crossover operators in NSGA-III for large-scale optimization problems. *Inf. Sci.* **2020**, *509*, 470–487. [CrossRef]
31. Yi, J.-H.; Deb, S.; Dong, J.; Alavi, A.H.; Wang, G. An improved NSGA-III algorithm with adaptive mutation operator for Big Data optimization problems. *Future Gener. Comput. Syst.* **2018**, *88*, 571–585. [CrossRef]
32. Liu, G.; Deng, M. Parametric active contour based on sparse decomposition for multi-objects extraction. *Signal Process.* **2018**, *148*, 314–321. [CrossRef]
33. Sun, J.; Miao, Z.; Gong, D.; Zeng, X.-J.; Li, J.; Wang, G.-G. Interval multi-objective optimization with memetic algorithms. *IEEE Trans. Cybern.* **2020**, *50*, 3444–3457. [CrossRef] [PubMed]
34. Zhang, Y.; Wang, G.; Li, K.; Yeh, W.-C.; Jian, M.; Dong, J. Enhancing MOEA/D with information feedback models for large-scale many-objective optimization. *Inf. Sci.* **2020**, *522*, 1–16. [CrossRef]
35. Srikanth, K.; Panwar, L.K.; Panigrahi, B.; Herrera-Viedma, E.; Sangaiah, A.K.; Wang, G. Meta-heuristic framework: Quantum inspired binary grey wolf optimizer for unit commitment problem. *Comput. Electr. Eng.* **2018**, *70*, 243–260. [CrossRef]
36. Chen, S.; Chen, R.; Wang, G.; Gao, J.; Sangaiah, A.K. An adaptive large neighborhood search heuristic for dynamic vehicle routing problems. *Comput. Electr. Eng.* **2018**, *67*, 596–607. [CrossRef]
37. Feng, Y.; Wang, G.-G. Binary moth search algorithm for discounted {0–1} knapsack problem. *IEEE Access* **2018**, *6*, 10708–10719. [CrossRef]
38. Feng, Y.; Wang, G.; Wang, L. Solving randomized time-varying knapsack problems by a novel global firefly algorithm. *Eng. Comput.* **2018**, *34*, 621–635. [CrossRef]
39. Abdel-Basst, M.; Zhou, Y. An elite opposition-flower pollination algorithm for a 0–1 knapsack problem. *Int. J. Bio-Inspired Comput.* **2018**, *11*, 46–53. [CrossRef]
40. Yi, J.-H.; Wang, J.; Wang, G. Improved probabilistic neural networks with self-adaptive strategies for transformer fault diagnosis problem. *Adv. Mech. Eng.* **2016**, *8*, 1–13. [CrossRef]
41. Mao, W.; He, J.; Li, Y.; Yan, Y. Bearing fault diagnosis with auto-encoder extreme learning machine: A comparative study. *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.* **2016**, *231*, 1560–1578. [CrossRef]
42. Mao, W.; Feng, W.; Liang, X. A novel deep output kernel learning method for bearing fault structural diagnosis. *Mech. Syst. Signal Process.* **2019**, *117*, 293–318. [CrossRef]
43. Duan, H.; Zhao, W.; Wang, G.; Feng, X. Test-Sheet Composition Using Analytic Hierarchy Process and Hybrid Metaheuristic Algorithm TS/BBO. *Math. Probl. Eng.* **2012**, *2012*, 1–22. [CrossRef]
44. Goldberg, D.E. Genetic algorithms in search. In *Optimization, and Machine Learning*; Addison-Wesley: Reading, MA, USA, 1989.

45. Mistry, K.; Zhang, L.; Neoh, S.C.; Lim, C.P.; Fielding, B. A Micro-GA Embedded PSO Feature Selection Approach to Intelligent Facial Emotion Recognition. *IEEE Trans. Cybern.* **2017**, *47*, 1496–1509. [CrossRef] [PubMed]
46. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
47. Song, X.-F.; Zhang, Y.; Guo, Y.-N.; Sun, X.-Y.; Wang, Y.-L. Variable-size Cooperative Coevolutionary Particle Swarm Optimization for Feature Selection on High-dimensional Data. *IEEE Trans. Evol. Comput.* **2020**, 1. [CrossRef]
48. Sun, Y.; Jiao, L.; Deng, X.; Wang, R. Dynamic network structured immune particle swarm optimisation with small-world topology. *Int. J. Bio-Inspired Comput.* **2017**, *9*, 93–105. [CrossRef]
49. Gandomi, A.H.; Alavi, A.H. Krill herd: A new bio-inspired optimization algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **2012**, *17*, 4831–4845. [CrossRef]
50. Abualigah, L.M.; Khader, A.T.; Hanandeh, E.S.; Gandomi, A.H. A novel hybridization strategy for krill herd algorithm applied to clustering techniques. *Appl. Soft Comput.* **2017**, *60*, 423–435. [CrossRef]
51. Wang, G.; Gandomi, A.H.; Alavi, A.H. An effective krill herd algorithm with migration operator in biogeography-based optimization. *Appl. Math. Model.* **2014**, *38*, 2454–2462. [CrossRef]
52. Wang, G.-G.; Gandomi, A.H.; Alavi, A.H.; Deb, S. A Multi-Stage Krill Herd Algorithm for Global Numerical Optimization. *Int. J. Artif. Intell. Tools* **2016**, *25*, 1550030. [CrossRef]
53. Wang, G.; Gandomi, A.H.; Alavi, A.H.; Gong, D. A comprehensive review of krill herd algorithm: Variants, hybrids and applications. *Artif. Intell. Rev.* **2019**, *51*, 119–148. [CrossRef]
54. Wang, G.; Guo, L.; Gandomi, A.H.; Hao, G.-S.; Wang, H. Chaotic Krill Herd algorithm. *Inf. Sci.* **2014**, *274*, 17–34. [CrossRef]
55. Dorigo, M.; Maniezzo, V.; Colorni, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man. Cybern. Part B Cybern.* **1996**, *26*, 29–41. [CrossRef]
56. Zheng, F.; Zecchin, A.C.; Newman, J.P.; Maier, H.R.; Dandy, G.C. An Adaptive Convergence-Trajectory Controlled Ant Colony Optimization Algorithm With Application to Water Distribution System Design Problems. *IEEE Trans. Evol. Comput.* **2017**, *21*, 773–791. [CrossRef]
57. Dorigo, M.; Stutzle, T. *Ant Colony Optimization*; MIT Press: Cambridge, UK, 2004.
58. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
59. Gao, S.; Yu, Y.; Wang, Y.; Wang, J.; Cheng, J.; Zhou, M. Chaotic Local Search-Based Differential Evolution Algorithms for Optimization. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, 1–14. [CrossRef]
60. Dulebenets, M.A. An Adaptive Island Evolutionary Algorithm for the berth scheduling problem. *Memetic Comput.* **2020**, *12*, 51–72. [CrossRef]
61. Dulebenets, M.A. A Delayed Start Parallel Evolutionary Algorithm for just-in-time truck scheduling at a cross-docking facility. *Int. J. Prod. Econ.* **2019**, *212*, 236–258. [CrossRef]
62. Agapitos, A.; Loughran, R.; Nicolau, M.; Lucas, S.; OrNeill, M.; Brabazon, A. A Survey of Statistical Machine Learning Elements in Genetic Programming. *IEEE Trans. Evol. Comput.* **2019**, *23*, 1029–1048. [CrossRef]
63. Back, T. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*; Oxford University Press: Oxford, UK, 1996.
64. Kashan, A.H.; Keshmiry, M.; Dahooie, J.H.; Abbasi-Pooya, A. A simple yet effective grouping evolutionary strategy (GES) algorithm for scheduling parallel machines. *Neural Comput. Appl.* **2016**, *30*, 1925–1938. [CrossRef]
65. Wang, G.; Gandomi, A.H.; Alavi, A.H.; Deb, S. A hybrid method based on krill herd and quantum-behaved particle swarm optimization. *Neural Comput. Appl.* **2015**, *27*, 989–1006. [CrossRef]
66. Wang, G.; Guo, L.; Wang, H.; Duan, H.; Liu, L.; Li, J. Incorporating mutation scheme into krill herd algorithm for global numerical optimization. *Neural Comput. Appl.* **2012**, *24*, 853–871. [CrossRef]
67. Abualigah, L.M.; Khader, A.T.; Hanandeh, E.S. A combination of objective functions and hybrid Krill herd algorithm for text document clustering analysis. *Eng. Appl. Artif. Intell.* **2018**, *73*, 111–125. [CrossRef]
68. Niu, P.; Chen, K.; Ma, Y.; Li, X.; Liu, A.; Li, G. Model turbine heat rate by fast learning network with tuning based on ameliorated krill herd algorithm. *Knowl. Based Syst.* **2017**, *118*, 80–92. [CrossRef]
69. Slowik, A.; Kwasnicka, H. Nature Inspired Methods and Their Industry Applications—Swarm Intelligence Algorithms. *IEEE Trans. Ind. Inform.* **2018**, *14*, 1004–1015. [CrossRef]

70. Brezočnik, L.; Fister, J.I.; Podgorelec, V. Swarm Intelligence Algorithms for Feature Selection: A Review. *Appl. Sci.* **2018**, *8*, 1521. [CrossRef]
71. Sun, C.; Jin, Y.; Cheng, R.; Ding, J.; Zeng, J. Surrogate-Assisted Cooperative Swarm Optimization of High-Dimensional Expensive Problems. *IEEE Trans. Evol. Comput.* **2017**, *21*, 644–660. [CrossRef]
72. Tran, B.N.; Xue, B.; Zhang, M. Variable-Length Particle Swarm Optimization for Feature Selection on High-Dimensional Classification. *IEEE Trans. Evol. Comput.* **2019**, *23*, 473–487. [CrossRef]
73. Tran, B.N.; Xue, B.; Zhang, M. A New Representation in PSO for Discretization-Based Feature Selection. *IEEE Trans. Cybern.* **2018**, *48*, 1733–1746. [CrossRef]
74. Zhang, J.; Zhu, X.; Wang, Y.; Zhou, M. Dual-Environmental Particle Swarm Optimizer in Noisy and Noise-Free Environments. *IEEE Trans. Cybern.* **2019**, *49*, 2011–2021. [CrossRef]
75. Bornemann, F.; Laurie, D.; Wagon, S.; Waldvogel, J. The siam 100-digit challenge: A study in high-accuracy numerical computing. *SIAM Rev.* **2005**, *1*, 47.
76. Epstein, A.; Ergezer, M.; Marshall, I.; Shue, W. Gade with fitness-based opposition and tidal mutation for solving ieee cec2019 100-digit challenge. In Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 10–13 June 2019; pp. 395–402.
77. Brest, J.; Maučec, M.S.; Bošković, B. The 100-digit challenge: Algorithm jde100. In Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 10–13 June 2019; pp. 19–26.
78. Zhang, S.X.; Chan, W.S.; Tang, K.S.; Zheng, S.Y. Restart based collective information powered differential evolution for solving the 100-digit challenge on single objective numerical optimization. In Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 10–13 June 2019; pp. 14–18.
79. Jun, S.; Bin, F.; Wenbo, X. Particle swarm optimization with particles having quantum behavior. In Proceedings of the 2004 Congress on Evolutionary Computation, Portland, OR, USA, 19–23 June 2004; Volume 321, pp. 325–331.
80. Wang, G.; Gandomi, A.H.; Alavi, A.H.; Hao, G.-S. Hybrid krill herd algorithm with differential evolution for global numerical optimization. *Neural Comput. Appl.* **2014**, *25*, 297–308. [CrossRef]
81. Wang, G.; Gandomi, A.H.; Alavi, A.H. Stud krill herd algorithm. *Neurocomputing* **2014**, *128*, 363–370. [CrossRef]
82. Wang, G.; Deb, S.; Gandomi, A.H.; Zhang, Z.; Alavi, A.H. Chaotic cuckoo search. *Soft Comput.* **2016**, *20*, 3349–3362. [CrossRef]
83. Wang, G.; Lu, M.; Zhao, X. An improved bat algorithm with variable neighborhood search for global optimization. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 1773–1778.
84. Simon, D. Biogeography-based optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [CrossRef]
85. Wang, G.; Tan, Y. Improving Metaheuristic Algorithms With Information Feedback Models. *IEEE Trans. Cybern.* **2017**, *49*, 542–555. [CrossRef]
86. Yang, X.-S.; Gandomi, A.H. Bat algorithm: A novel approach for global engineering optimization. *Eng. Comput.* **2012**, *29*, 464–483. [CrossRef]
87. Wang, G.; Guo, L. A Novel Hybrid Bat Algorithm with Harmony Search for Global Numerical Optimization. *J. Appl. Math.* **2013**, *2013*, 1–21. [CrossRef]
88. Wang, G.; Guo, L.; Duan, H.; Liu, L.; Wang, H. Dynamic Deployment of Wireless Sensor Networks by Biogeography Based Optimization Algorithm. *J. Sens. Actuator Netw.* **2012**, *1*, 86–96. [CrossRef]
89. Li, J.; Li, Y.-X.; Tian, S.-S.; Zou, J. Dynamic cuckoo search algorithm based on taguchi opposition-based search. *Int. J. Bio-Inspired Comput.* **2019**, *13*, 59–69. [CrossRef]
90. Yang, X.-S.; Deb, S. Engineering optimisation by cuckoo search. *Int. J. Math. Model. Numer. Optim.* **2010**, *1*, 330. [CrossRef]
91. Gandomi, A.H.; Yang, X.-S.; Alavi, A.H. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Eng. Comput.* **2013**, *29*, 17–35. [CrossRef]
92. Wang, G.; Gandomi, A.H.; Zhao, X.; Chu, H.C.E. Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. *Soft Comput.* **2016**, *20*, 273–285. [CrossRef]
93. Wang, G.; Deb, S.; Coelho, L.D.S. Earthworm optimization algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Int. J. Bio-Inspired Comput.* **2018**, *12*, 1–22. [CrossRef]

94. Wang, G.-G.; Deb, S.; Coelho, L.D.S. Elephant herding optimization. In Proceedings of the 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI 2015), Bali, Indonesia, 7–9 December 2015; pp. 1–5.
95. Wang, G.-G.; Deb, S.; Gao, X.-Z.; Coelho, L.d.S. A new metaheuristic optimization algorithm motivated by elephant herding behavior. *Int. J. Bio-Inspired Comput.* **2016**, *8*, 394–409. [CrossRef]
96. Wang, G. Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Memetic Comput.* **2018**, *10*, 151–164. [CrossRef]
97. Yang, X.-S. Firefly algorithm, stochastic test functions and design optimisation. *Int. J. Bio-Inspired Comput.* **2010**, *2*, 78. [CrossRef]
98. Wang, H.; Yi, J.-H. An improved optimization method based on krill herd and artificial bee colony with information exchange. *Memetic Comput.* **2018**, *10*, 177–198. [CrossRef]
99. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
100. Liu, F.; Sun, Y.; Wang, G.-G.; Wu, T. An artificial bee colony algorithm based on dynamic penalty and chaos search for constrained optimization problems. *Arab. J. Sci. Eng.* **2018**, *43*, 7189–7208. [CrossRef]
101. Geem, Z.W.; Kim, J.H.; Loganathan, G. A New Heuristic Optimization Algorithm: Harmony Search. *Simulation* **2001**, *76*, 60–68. [CrossRef]
102. Wang, G.; Deb, S.; Cui, Z. Monarch butterfly optimization. *Neural Comput. Appl.* **2019**, *31*, 1995–2014. [CrossRef]
103. Wang, G.; Deb, S.; Zhao, X.; Cui, Z. A new monarch butterfly optimization with an improved crossover operator. *Oper. Res.* **2018**, *18*, 731–755. [CrossRef]
104. Gandomi, A.H.; Alavi, A.H. Multi-stage genetic programming: A new strategy to nonlinear system modeling. *Inf. Sci.* **2011**, *181*, 5227–5239. [CrossRef]

mathematics

MDPI

Article

Elephant Herding Optimization: Variants, Hybrids, and Applications

Juan Li [1,2,3], Hong Lei [2], Amir H. Alavi [4,5,6] and Gai-Ge Wang [7,8,9,*]

1 School of Artificial Intelligence, Wuhan Technology and Business University, Wuhan 430065, China; looj@whu.edu.cn
2 School of Artificial Intelligence, Wuchang University of Technology, Wuhan 430223, China; lh120150508@wut.edu.cn
3 Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China
4 Department of Civil and Environmental Engineering, University of Pittsburgh, Pittsburgh, PA 15261, USA; alavi@pitt.edu
5 Department of Bioengineering, University of Pittsburgh, Pittsburgh, PA 15261, USA
6 Department of Computer Science and Information Engineering, Asia University, Taichung 41354, Taiwan
7 Department of Computer Science and Technology, Ocean University of China, Qingdao 266100, China
8 Institute of Algorithm and Big Data Analysis, Northeast Normal University, Changchun 130117, China
9 School of Computer Science and Information Technology, Northeast Normal University, Changchun 130117, China
* Correspondence: wgg@ouc.edu.cn

Received: 8 July 2020; Accepted: 20 August 2020; Published: 24 August 2020

Abstract: Elephant herding optimization (EHO) is a nature-inspired metaheuristic optimization algorithm based on the herding behavior of elephants. EHO uses a clan operator to update the distance of the elephants in each clan with respect to the position of a matriarch elephant. The superiority of the EHO method to several state-of-the-art metaheuristic algorithms has been demonstrated for many benchmark problems and in various application areas. A comprehensive review for the EHO-based algorithms and their applications are presented in this paper. Various aspects of the EHO variants for continuous optimization, combinatorial optimization, constrained optimization, and multi-objective optimization are reviewed. Future directions for research in the area of EHO are further discussed.

Keywords: elephant herding optimization; engineering optimization; metaheuristic; constrained optimization; multi-objective optimization

1. Introduction

The rapid growth of the size and complexity of optimization problems implies that the traditional optimization algorithms are becoming more uncertain for solving these problems [1]. Metaheuristic algorithms [2–4] have proved to be a viable solution to this challenge. Inspired by nature, these strong metaheuristic algorithms are applied to solve NP-hard problems, such as flow shop scheduling [5–9], image encryption [10–12], feature selection [13–15], facial feature detection [16,17], path planning [18,19], cyber-physical social systems [20,21], texture discrimination [22], factor evaluation [23], saliency detection [24], classification [25], engineering optimization [26], object extraction [27], gesture segmentation [28], economic load dispatch [29], shape design [30], big data and large-scale optimization [31], signal processing [32], multi-objective and many-objective optimization [33–35], unit commitment [36], vehicle routing [37,38], and the knapsack problem [39,40]. Some of the well-known methods in this area are genetic algorithms (GAs) [41], particle swarm optimization (PSO) [42–45], differential evolution (DE) [19,46,47], monarch butterfly optimization (MBO) [48–52], artificial bee colonies (ABCs) [53], earthworm optimization algorithms (EWAs) [54], ant colony optimization

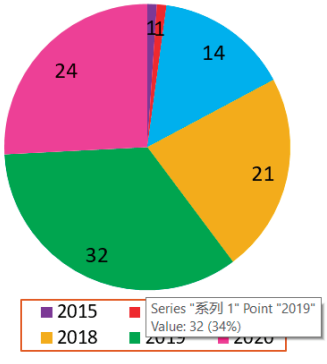
(ACO) [55], cuckoo search (CS) [56–62], krill herd (KH) [63–67], firefly algorithms (FAs) [68–73], simulated annealing (SA) [74], intelligent water drop (IWD) [75], water cycle algorithms (WCAs) [76], moth search (MS) [77], monkey algorithms (MAs) [78], evolutionary strategy (ES) [79], free search (FS) [80], probability-based incremental learning (PBIL) [81], biogeography-based optimization (BBO) [82–85], dragonfly algorithms (DAs) [86], interior search algorithms (ISAs) [87], brain storm optimization (BSO) [88,89], bat algorithms (BAs) [18,90–97], stud GAs (SGAs) [98], harmony search (HS) [99–102], fireworks algorithms (FWAs) [103], and chicken swarm optimization (CSO) [104].

Based on the herding behavior of elephants, a new swarm intelligence-based global optimization algorithm, namely elephant herding optimization (EHO), was proposed by Wang et al. [105]. Two special operators, a clan updating operator and a separating operator, are included in EHO. The elephants in each clan are updated with respect to their current position and the position of the matriarch. The acceptable performance of EHO has drawn much attention from scholars and engineers. In this paper, a comprehensive review for the EHO-based algorithms and their applications are presented. The remainder of this paper is organized as follows. The main steps of the EHO is detailed in Section 2. Improved EHO algorithm variants are presented in Section 3. Section 4 describes the EHO applications for solving engineering optimization problems. Finally, Section 5 presents a conclusion and suggestions for future work.

2. Historical Development of Elephant Herding Optimization

2.1. Elephant Herding Optimization Research Studies

The EHO algorithm with the herding behavior of elephant groups has received significant attention from scholars [105]. Ninety-three related studies have been published in journals/dissertations/conferences up to 23 April 2020 (Figure 1) since EHO was proposed in 2015. Among these 93 papers, 2 papers were published in 2015 and 2016, 14 papers were published in 2017, 21 papers were published in 2018, and 32 and 24 papers were published in 2019 and 2020, respectively.



Year	Number	%
2015	1	1%
2016	1	1%
2017	14	15%
2018	21	23%
2019	32	34%
2020	24	26%

Figure 1. Related (Elephant Herding Optimization) EHO publications since 2015.

2.2. Basics of Elephant Herding Optimization

Elephants, as social creatures, live in social structures of females and calves. An elephant clan is headed by a matriarch and composed of a number of elephants. Female members like to live with family members, while the male members tend to live elsewhere. They will gradually become independent of their families until they leave their families completely. The population of all elephants is shown in Figure 2. The EHO technique proposed by Wang et al. in 2015 [105] was developed after studying natural elephant herding behavior. The following assumptions are considered in EHO.

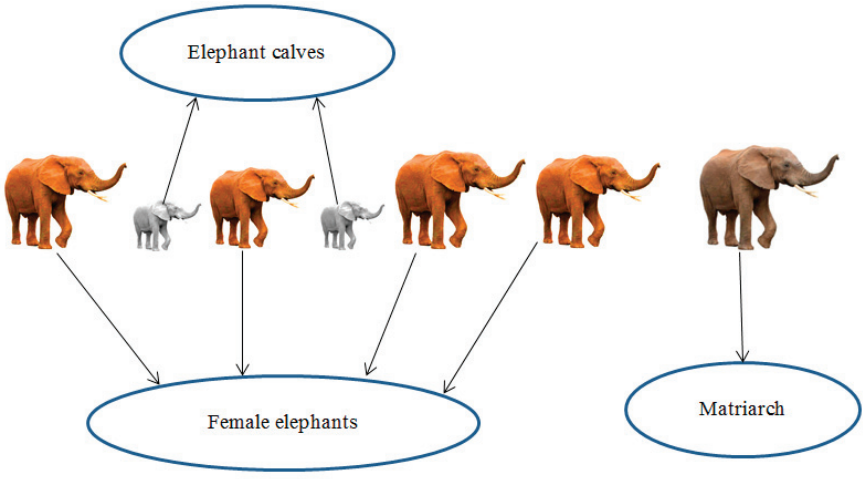


Figure 2. Population of elephants.

(1) Some clans with fixed numbers of elephants comprise the elephant population.

(2) A fixed number of male elephants will leave their family group and live solitarily far away from the main elephant group in each generation.

(3) A matriarch leads the elephants in each clan.

2.2.1. Clan-updating Operator

According to the natural habits of elephants, a matriarch leads the elephants in each clan. Therefore, the new position of each elephant *ci* is influenced by matriarch *ci*. The elephant *j* in clan *ci* can be calculated by Equation (1).

$$x_{new,ci,j} = x_{ci,j} + a \times (x_{best,ci} - x_{ci,j}) \times r \tag{1}$$

where $x_{new,ci,j}$ and $x_{ci,j}$ present the new position and old position for elephant *j* in clan *ci*, respectively. $x_{best,ci}$ is matriarch c_i which represents the best elephant in the clan. $a \in [0,1]$ indicates a scale factor, $r \in [0,1]$. The best elephant can be calculated by Equation (2) for each clan.

$$x_{new,ci,j} = \beta \times x_{center,ci} \tag{2}$$

where $\beta \in [0,1]$ represents a factor which determines the influence of the $x_{center,ci}$ on $x_{new,ci,j}$. $x_{new,ci,j}$ is the new individual. $x_{center,ci}$ is the center individual of clan *ci*. It can be calculated by Equation (3) for the *d*-th dimension.

$$x_{center,ci,d} = \frac{1}{n_{ci}} \times \sum_{j=1}^{n_{ci}} x_{ci,j,d} \tag{3}$$

where $1 \le d \le D$ and n_{ci} indicate the number of elephants in clan *ci*. $x_{ci,j,d}$ represents the *d*-th dimension of elephant individual $x_{ci,j}$. $x_{center,ci}$ is the center of clan *ci* and it can be updated by Equation (3).

2.2.2. Separating Operator

The separating process whereby male elephants leave their family group can be modeled into separating operator when solving optimization problems. The separating operator is implemented by the elephant individual with the worst fitness in each generation, as shown in Equation (4).

$$x_{worst,ci} = x_{\min} + (x_{\max} - x_{\min} + 1) \times rand \tag{4}$$

where x_{max} represents the upper bound of the individual and x_{min} indicates lower bound of the individual. $x_{worst,ci}$ indicates the worst individual in clan c_i. *Rand* [0, 1] is a stochastic distribution between 0 and 1.

According to the description of the clan-updating operator and separating operator, the mainframe of EHO is summarized. The corresponding flowchart is shown as follows. *MaxGen* is the maximum generation. The MATLAB code of EHO can be found on the website: https://www.mathworks.com/matlabcentral/fileexchange/53486. The basic steps of the EHO is shown as follows (Algorithm 1). The corresponding flowchart can also be seen in Figure 3.

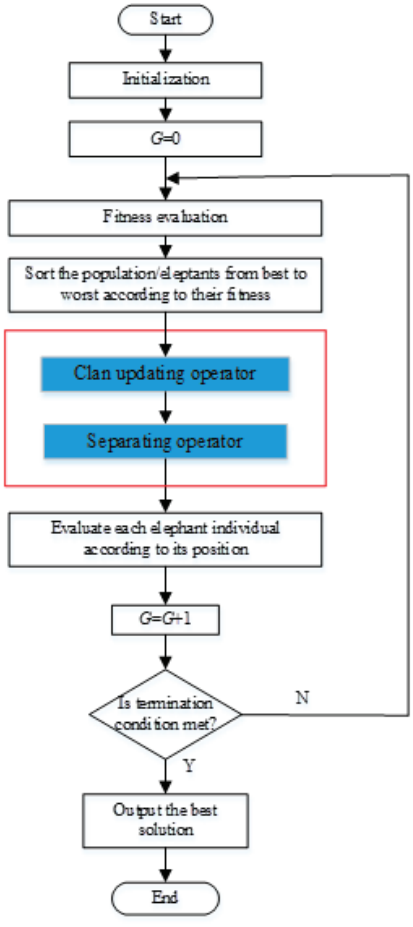


Figure 3. Flowchart of the EHO algorithm.

Algorithm 1. Elephant herding optimization

(1) **Begin**
(2) **Initialization.** Set the initialize iterations $G = 1$; initialize the population P randomly; set maximum generation *MaxGen*.
(3) **While** stopping criterion is not met **do**
(4) Sort the population according to fitness of individuals.
(5) **For** all clans *ci* do
(6) **For** elephant *j* in the clan *ci* **do**
(7) Generate $x_{new,\,ci,j}$ and update $x_{ci,j}$ by Equation (1).
(8) **If** $x_{ci,j} = x_{best,ci}$ **then**
(9) Generate $x_{new,\,ci,j}$ and update $x_{ci,j}$ by Equation (2).
(10) **End if**
(11) **End for**
(12) **End for**
(13) **For** all clans *ci* **do**
(14) Replace the worst individual *ci* by Equation (4).
(15) **End for**
(16) Evaluate each elephant individual according to its position.
(17) $T = T + 1$.
(18) **End while**
(19) **End.**

2.2.3. Analysis of Algorithm Complexity

The computational complexity of the EHO algorithm is analyzed according to the steps in the EHO algorithm. Let the population size and dimension be NP and D, respectively. Obviously, sort the population according to the fitness of individuals in step (4) with time complexity $O(NP)$. In steps (5)–(12), execute clan-updating operator for all clans *ci* with time complexity $O(NP \times D)$. In steps (13)–(15), execute separating operator for all clans *ci* with time complexity $O(NP)$. Evaluate each elephant individual according to its position in step (16) with time complexity $O(NP)$. To do so, the total time complexity of elephant herding optimization is $O(T \times NP \times D)$. From the above results, after omitting the low-order terms, the total time complexity of the EHO algorithm is $O(T \times NP \times D)$, which is only related to T, NP, and D.

3. Different Variants of EHO

Several EHO variants have been proposed to solve different optimization problems. The variants of EHO can be generally divided into three groups: improved EHO algorithms, hybrid EHO algorithms, and variants of EHO.

3.1. Improved EHO Algorithms

A list of the improved EHO algorithms is given in Table 1 and Figure 4. An overview of each of these methods is given below.

3.1.1. Chaotic EHO

Tuba et al. [106] proposed a new EHO algorithm with chaos theory called CEHO to solve unconstrained global optimization problems. In CEHO, two different chaotic maps are introduced into the EHO algorithm. Compared with 15 standard benchmark functions from IEEE Congress on Evolutionary Computation (CEC) 2013, the CEHO algorithm outperforms the basic EHO and PSO in almost all cases.

Table 1. The improved EHO algorithms.

Name	Author	Reference
Chaotic elephant herding optimization (CEHO)	Tuba et al.	[106]
EHO with individual updating strategies	Li et al.	[107]
EHO with Lévy flight (LFEHO)	Xu et al.	[108]
Improved elephant herding optimization (IEHO)	Xu et al.	[109]
Multi-search elephant herding optimization (Multi-EHO)	Hakli et al.	[110]
k-means EHO	Tuba et al.	[111]
Dynamic Cauchy mutation EHO (EHO-DCM)	Chakraborty et al.	[112]
Adaptive whale elephant herding optimization (AWEHO)	Chowdary et al.	[113]

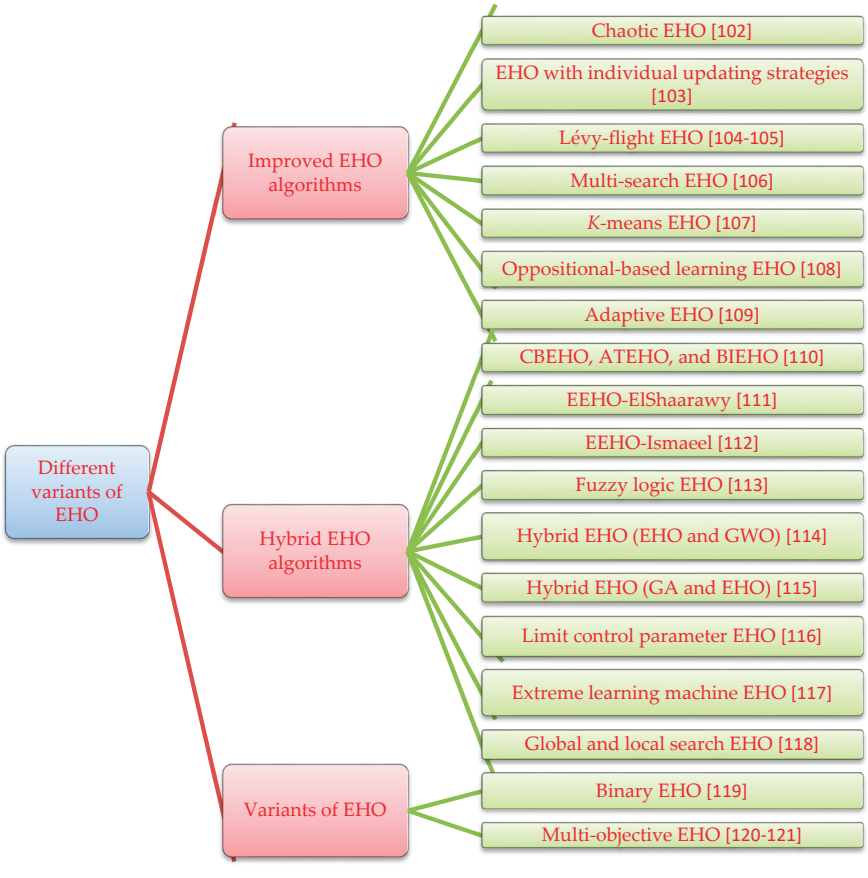


Figure 4. Different variants of EHO.

3.1.2. EHO with Individual Updating Strategies

Li et al. [107] incorporated six individual updating strategies into basic EHO. The experimental results for sixteen test functions show that the proposed improved EHO variant significantly outperformed basic EHO.

3.1.3. Lévy Flight EHO

Xu et al. [108] applied an improved EHO algorithm with Lévy flight (LFEHO) to solve network intrusion detection problems. The research results showed that the LFEHO algorithm increased the accuracy rate of the network.

Xu et al. [109] proposed improved EHO (IEHO) to solve network intrusion detection problems, which improved the classification performance of intrusion detection under the premise of ensuring the accuracy rate and meeting the needs in real time. The experimental results showed that the IEHO algorithm was superior to other algorithms (EHO [105], PSO [2], and MS [77]).

3.1.4. Multi-Search EHO

Hakli et al. [110] proposed new EHO with a multi-search strategy (multi-EHO). Fifteen different benchmark functions were used to verify the effectiveness of multi-EHO. In addition, the proposed multi-EHO was compared with the whale optimization algorithm (WOA) and the gray wolf optimizer (GWO). The multi-EHO method was also superior to other competitive algorithms.

3.1.5. k-Means EHO

Tuba et al. [111] introduced data clustering into EHO in which the local search ability of EHO was improved through *k*-means. The proposed *k*-means EHO was tested on six benchmark datasets. The clustering results showed that *k*-means EHO found better clusters than other algorithms.

3.1.6. Oppositional-Based Learning EHO

Chakraborty et al. [112] proposed improved EHO with a dynamic Cauchy mutation (EHO-DCM) to solve the multilevel image thresholding for image segmentation problems. In EHO-DCM, oppositional-based learning (OBL) and DCM were introduced, in which OBL and DCM were employed to accelerate the conventional and mitigate the premature convergence, respectively. The results were compared with five metaheuristic algorithms (EHO [105], CSO [104], ABCs [53], BAs [91], and PSO [2]). It was demonstrated that EHO-DCM provided promising performance in view of optimized fitness value, feature similarity index, and structure similarity index.

3.1.7. Adaptive Whale EHO

Chowdary et al. [113] proposed a hybrid mixture model based on the adaptive whale EHO (AWEHO) algorithm, which is the integration of three technologies: EHO [105], the whale optimization algorithm (WOA), and the adaptive concept. In the proposed method, the AWEHO algorithm was applied to perform optimal sensing by using the foraging behavior of whales and the herding behavior of elephants. The analysis of the computational results indicated that the herding and foraging behavior of the AWEHO achieved efficient spectrum sensing in the cognitive radio network.

3.2. Hybrid EHO Algorithms

The hybrid EHO algorithms are presented in Table 2. The details are included in the following sections.

3.2.1. CBEHO, ATEHO, and BIEHO

Rashwan et al. [114] studied three approaches, which are cultural-based EHO (CBEHO), alpha-tuning EHO (ATEHO), and biased initialization EHO (BIEHO), to enhance the performance of standard EHO. A comparative experiment from CEC 2016 was done between three EHO approaches and the other optimization methods. It was demonstrated that the performances of the three EHO approaches were superior to other comparison methods. In order to further verify the performance of the three EHO approaches, various experiments were carried out on engineering problems such as the welded beam, gear train, continuous stirred tank reactor, and three-bar truss design problem.

Table 2. The hybrid EHO algorithms.

Name	Author	Reference
Cultural-based EHO, alpha-tuning EHO, and biased initialization EHO (CBEHO, ATEHO, and BIEHO)	Rashwan et al.	[114]
Enhanced elephant herding optimization (EEHO-ElShaarawy)	ElShaarawy et al.	[115]
Enhanced elephant herding optimization (EEHO-Ismaeel)	Ismaeel et al.	[116]
Fuzzy elephant herding optimization (FEHO)	Veera et al.	[117]
Elephant herding optimization and gray wolf optimization (EHGWO)	Arora et al.	[118]
Genetic algorithm and elephant herding optimization (GEHO)	Bukhsh et al.	[119]
Hybrid elephant herding optimization (HEHO)	Ivana et al.	[120]
Extreme learning machine and elephant herding optimization (ELM-EHO)	Satapathy et al.	[121]
Global and local search (GL-EHO)	Hakli et al.	[122]

3.2.2. EEHO-ElShaarawy

ElShaarawy et al. [115] used an enhanced elephant herding optimization (EEHO-ElShaarawy) algorithm to overcome the fast convergence of EHO. The exploitation and exploration of EEHO-ElShaarawy were achieved by separating operators with balanced control. EEHO-ElShaarawy showed a better performance of convergence rate compared with basic EHO.

3.2.3. EEHO-Ismaeel

Ismaeel et al. [116] proposed another enhanced elephant herding optimization algorithm with a constant function (EEHO-Ismaeel). In order to overcome the shortcomings of EHO. In EEHO-Ismaeel, two operators, the clan and separating operator, improved the exploitation abilities of the EEHO-Ismaeel algorithms. The CEC 2017 test benchmark functions were utilized to verify the performance of the three EEHO-Ismaeel versions (EEHO15, EEHO20, and EEHO25). The experimental results demonstrated that, in most cases, the EEHO-Ismaeel algorithms obtained better results compared with the other competitive algorithms, such as the PSO, bird swarm algorithm (BSA), and ant lion optimization (ALO) algorithms.

3.2.4. FEHO

Veera et al. [117] introduced a fuzzy logic controller into EHO and proposed improved fuzzy EHO (FEHO) to maximize power point tracking (MPPT) for a hybrid wind–solar system. Simulation results indicated that the MPPT using the proposed FEHO had better performance compared with the other type of controllers, which efficiently tracked the maximum power point of the wind–solar power systems even with variations in the climatic conditions.

3.2.5. EHGWO

Arora et al. [118] combined the advantages of EHO and GWO and proposed a hybrid algorithm (EHGWO). In EHGWO, the optimal virtual machines (VMs) are selected and reallocated by using a newly devised fitness function. The tasks for overloaded VMs are removed and assigned to VMs without affecting the system performance, which performed a load balancing technique.

3.2.6. GEHO

Bukhsh et al. [119] proposed a hybrid algorithm called GEHO by combining a GA and EHO. Based on the results, the developed GEHO approach was able to schedule the appliance efficiently, which reduced maximum cost compared with EHO for home appliance optimization problems.

3.2.7. HEHO

Strumberger et al. [120] developed improved hybrid EHO, named HEHO, to solve the wireless sensor network localization problem. The limit control parameter from the ABC algorithm was

incorporated into EHO to control the process of diversification. The usefulness of HEHO was demonstrated using different sizes of sensor networks from 25 to 150 target nodes. Based on the results, the HEHO approach was able to obtain more consistent and accurate locations of the unknown target nodes than other approaches.

3.2.8. ELM-EHO

Satapathy et al. [121] proposed a combination model named EHO-ELM with a combination of the advantages of extreme learning machine (ELM) and EHO. In this model, EHO-ELM was used to determine the input weights of an ELM model. EHO-ELM was tested on three different brain image datasets. The results demonstrated that EHO-ELM outperformed the basic ELM model in the three brain image datasets.

3.2.9. Global and Local Search EHO

Hakli et al. [122] developed a new EHO approach to solve constrained optimization problems. The EHO variants (GL-EHO) were adapted to implement constrained optimization. Experimental results showed that GL-EHO was capable of overtaking EHO.

3.3. Variants of EHO

Different variants of the EHO algorithm are presented in Table 3. The detailed methods are presented herein.

Table 3. Different variants of EHO.

Name	Author	Reference
Binary EHO algorithm (BinEHO)	Huseyin et al.	[123]
Multi-objective clustering EHO algorithm (MOEHO)	Jaiprakash et al.	[124]
Improved and multi-objective EHO (IMOEHO)	Meena et al.	[125]

3.3.1. Binary EHO

Hakli et al. [123] proposed a new binary variant of EHO (BinEHO) for solving binary optimization problems. Through a dimension rate (DR) parameter and mutation process, BinEHO strengthened the compromise between exploitation and exploration. In order to prove the robustness and accuracy of BinEHO, it was compared with various binary variants in three different binary optimization problems. The results concluded that BinEHO outperformed the other binary algorithm variants.

3.3.2. Multi-Objective EHO

Jaiprakash et al. [124] presented a multi-objective clustering EHO (MOEHO) to solve multi-objective optimization problems. Comparative results revealed that MOEHO provided superior performance compared with (fast and elitist multiobjective genetic algorithm) NSGA-II and MOPSO in eight cases. In addition, MOEHO was used to cluster the activities of human models. The results showed that MOEHO succeeded in eight out of five case studies.

Meena et al. [125] presented improved multi-objective EHO (IMOEHO) to solve distribution system optimization problems. In IMOEHO, two techniques (order of preference by similarity to the ideal solution technique and improved EHO technique) were combined. The IMOEHO method was implemented in three benchmark test distribution systems. It was concluded that the IMOEHO method was very effective for optimizing multi-objective complex optimization problems.

4. Engineering Optimization/Applications

The EHO algorithm has been successfully applied to engineering optimization problems since it was proposed. A summary for EHO in engineering optimization is presented in Tables 4 and 5 and Figure 5.

Table 4. A summary of the EHO applications in engineering optimization.

Category	Problem/Application	Author	Ref.
Continuous optimization	Training artificial neural networks	Moayedi et al.	[126]
	Selecting structure and weights for neural networks	Kowsalya et al.	[127]
	Training neural networks	Sahlol et al.	[128]
	Optimizing underwater sensor networks	Sukhman et al.	[129]
	Unmanned aerial vehicle path planning	Alihodzic et al.	[130]
	Clustering	Rani et al. Jaiprakash et al.	[131] [132]
	Support vector regression (SVR) classifier	Hassanien et al. Hassanien et al. Tuba et al. Tuba et al.	[133] [134] [135] [136]
	Control problem	Sambariya et al.	[137]

4.1. Continuous Optimization

4.1.1. Neural Networks

Moayedi et al. [126] synthesized a new EHO-MLP ensemble with a multi-layer perceptron (MLP) neural network to predict cooling load. The results revealed that EHO-MLP performed efficiently for adjusting biases of the MLP and the neural weights. It also outperformed the ACO [55] and EHO [105] optimization algorithms both in training and testing accuracies. Meanwhile, EHO-MLP took less time than ACO [55] and EHO [105] with regard to the time-effectiveness of the models.

Kowsalya et al. [127] used EHO to optimize neural network weights. The performance of the proposed method was evaluated on evaluation metrics. It was concluded that the proposed method provided better accuracy than existing classifiers.

Sahlol et al. [128] applied EHO to neural networks to classify each cell for the acute lymphoblastic leukemia problem. In the proposed method, the weights and biases of the network were updated by the EHO algorithm. The research results showed that EHO outperformed other classification methods.

4.1.2. Underwater Sensor Networks

Kaur et al. [129] used EHO to solve underwater sensor networks optimization tasks. The research outcomes indicated that the proposed approach showed better performance than other strategies for most parameters.

4.1.3. Unmanned Aerial Vehicle Path Planning

Alihodzic et al. [130] considered an approximation algorithm, adjusted EHO (AEHO), to solve the unmanned aerial vehicle (UAV) path planning problem. AEHO was used for adjusting the UAV path planning problem and it was compared with other state-of-the-art algorithms. The simulation experiments showed that AEHO obtained a safe flight path and was an excellent choice for the UAV path planning problem.

4.1.4. Clustering

Rani et al. [131] proposed a new detection approach for dynamic protein complexes by using Markov clustering with EHO (MC-EHO). The MC-EHO method divided the protein–protein interaction (PPI) network into a set of dynamic sub-networks and employed the clustering analysis on every sub-network. The experimental analysis was employed on 11 various widespread datasets and four different benchmark databases. The results showed that MC-EHO surpassed various existing approaches in terms of accuracy measures.

Jaiprakash et al. [132] formulated EHO to perform a clustering task by minimizing intra-cluster distance. The simulation was verified on six benchmark datasets and three synthetic datasets. The superior percentage accuracy of EHO was demonstrated by comparing it with other algorithms in the form of box plots.

Table 5. A summary of the EHO applications in engineering optimization.

Category	Problem/Application	Author	Ref.
Combinatorial optimization	Traveling salesman problem	Almufti et al.	[138]
	Knapsack	Darmawan et al.	[139]
	Acoustic energy-based positioning	Arora et al.	[140]
	Scheduling	Parasha et al. Cahig et al. Sarwar et al. Komal et al. Mohsin et al. Gholam et al. Fatima et al.	[141] [142] [143] [144] [145] [146] [147]
	Electrostatic powder coating process	Pongchanun et al.	[148]
	Image safety model	Shankar et al. Chibani et al.	[149] [150]
	Image processing	Tuba et al. Shankar et al. Jayanth et al. Cardoso et al.	[151] [152] [153] [154]
	Wireless sensor networks	Sérgio et al. Ivana et al. Kaur et al.	[155] [156] [157]
	Feature selection	Xu et al.	[158]
	Optimal power flow problem	Mukherjee et al. S. Mani et al. Sambariya et al.	[159] [160] [161]
	Distribution systems	Prasad et al. Vijay et al.	[162] [163]
Constrained Optimization	Linear and nonlinear constrained optimization problems	Ivana e et al.	[164]
	Economic dispatch problems	Singh et al.	[165]
	Stochastic inequality constrained optimization problems	Horng et al.	[166]
Multi-objective optimization	Quality of service (QoS) aware web service composition optimization	Sadouki et al.	[167]
	Civil engineering	Adarsha et al.	[168]
	Structural optimization	Malihe et al.	[169]

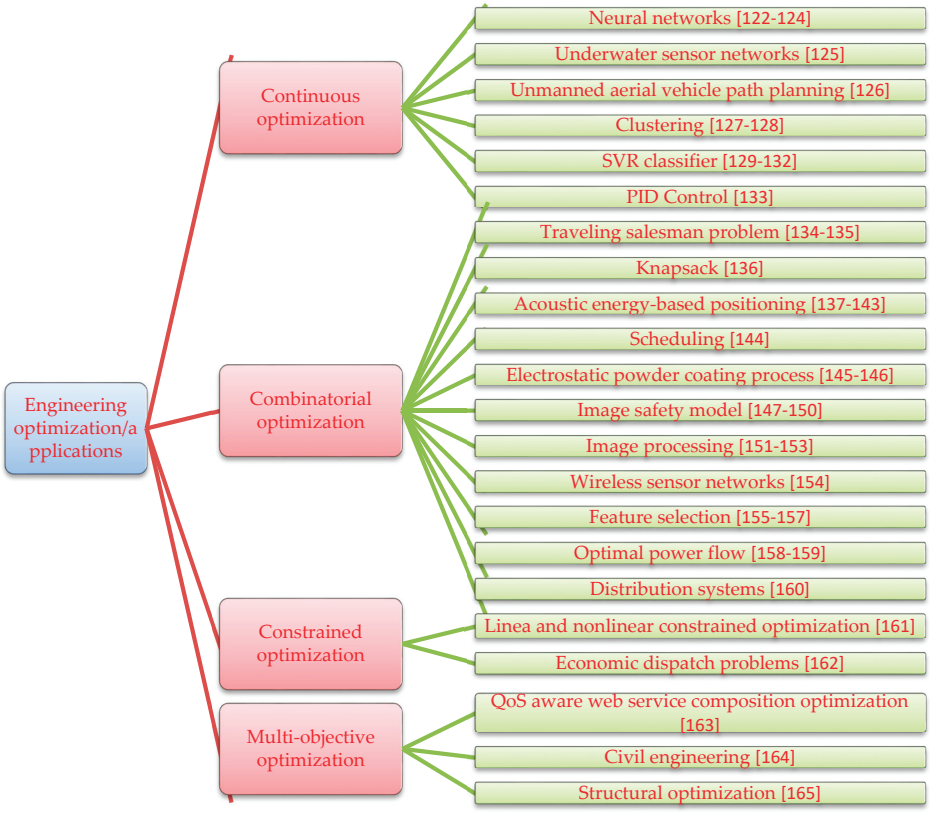


Figure 5. Engineering optimization/applications.

4.1.5. SVR Classifier

Hassanien et al. [133] introduced EHO to adjust the regression of emotional states for a support vector regression (SVR) repressor (SVR-EHO). In this method, the feature selection was adapted and the SVR classifier parameters were adjusted by using EHO, which provided a fast regression rate. The SVR-EHO approach was verified on the open database for emotion detection. The results of emotion regression on the SVR classifier indicated that SVR-EHO significantly improved regression accuracy.

Hassanien et al. [134] used two technologies, EHO and SVM (EHO-SVM), to develop a hybrid approach for automatic electrocardiogram (ECG) signal classification. The proposed approach included three modules, which were the efficient preprocessing module, feature extraction module, and feature classification module. EHO-SVM was utilized to optimize the features and parameters. The experiments showed that EHO-SVM achieved accurate classification results in terms of five statistical indices.

Tuba et al. [135] used the EHO algorithm to adjust the SVM parameter. The proposed approach was tested on standard datasets and the results were obtained by EHO and compared with two other approaches, which were the GA [41] and the grid search method (Grid). The computational experiments concluded that the EHO algorithm outperformed the GA [41] and Grid in the accuracy of classification for the same test problems.

Tuba et al. [136] used the EHO algorithm to find the optimal parameters of the SVM. In the proposed approach, the parameters of SVM were adjusted by EHO. Four different experiments based on a standard dataset were carried out. The simulation results showed that the performance of the proposed method achieved better results than the other strategies in all cases.

4.1.6. PID Control

Sambariya et al. [137] used the EHO algorithm to adjust the parameters of the proportional integral derivative (PID) controller, which minimized the change in frequency of a single-area non-reheat thermal power plant. The experimental results showed that a controller based on EHO had a better performance than other conventional PID controllers.

4.2. Combinatorial Optimization

4.2.1. Traveling Salesman Problem

Almufti et al. [138] introduced EHO to solve symmetric traveling salesman problems (STSPs). The experiment results indicated that EHO was adapted to solve STSPs by comparing the optimal solutions of the traveling salesman problem library (TSPLIB).

4.2.2. Knapsack

Darmawan et al. [139] used the EHO algorithm to solve 0–1 knapsack problems. The analysis of the computational results indicated that EHO outperformed other algorithms for convergence rate and global search ability when more and more iterations were done.

4.2.3. Acoustic Energy-Based Positioning

Correia et al. [140] used the EHO algorithm to validate and adjust the decay acoustic model for acoustic energy-based positioning problems. The implementation results for both simulation results and real measurements showed EHO had a good alignment with conducted simulations and was successfully applied to acoustic energy-based positioning problems.

4.2.4. Scheduling

Parashar et al. [141] used modified elephant herding optimization (MEHO) to model uncertain renewable generation. The analysis of the computational results indicated that the proposed MEHO approach had significant effects on the operational management of the microgrid compared with the deterministic approach.

Cahig et al. [142] proposed a decision tool based on EHO for a virtual power plant (VPP) scheduling problem. The algorithm was illustrated for a test system with a VPP. The results showed that the canonical variant of EHO yielded the optimal scheduling, which suggested that it performed well as a decision support tool to the VPP operator.

Sarwar et al. [143] used EHO to solve a home energy management system (HEMS) scheduling problem. Simulations of a single home with 12 appliances were performed and the results showed the EHO technique performed better than the other reported algorithms in reducing the waiting time and cost.

Parvez et al. [144] used two optimizing techniques, EHO [105] and harmony search algorithm (HSA) [99], to evaluate the performance of a home energy management system (HEMS). The simulation results revealed that the proposed method was more effective in terms of electricity cost.

Mohsin et al. [145] implemented the EHO technique to solve the scheduling of smart home appliances. The simulation results revealed that EHO performed much better in terms of total cost and peak load reduction for different operation time intervals (OTIs). In addition, EHO with shorter OTIs provided better results compared with longer OTIs.

Gholami et al. [146] developed improved EHO to solve large instances for hybrid flow shop scheduling problems. The performance of the proposed algorithm was compared with two available algorithms, which were SA and shuffled frog-leaping algorithm (SFLA). Based on the results, the developed approach outperformed the other algorithms.

Fatima et al. [147] developed an efficient optimization method via the hybridization of two optimization algorithms, namely EHO [105] and the FA [69]. This method was used to reduce the electricity cost for home energy management controller problems. The results indicated that the proposed hybrid optimization technique performed more efficiently for achieving the lowest cost and maximizing consumer satisfaction.

4.2.5. Electrostatic Powder Coating Process

Luangpaiboon et al. [148] proposed a modified simplex EHO algorithm with multiple performance measures (MEHO). MEHO was used to solve the optimization of electrostatic powder coating process parameter optimization problems. According to some performance measures, two phases based on the response surface methodology were applied to study the EHO parameter levels. The simulation experimental results demonstrated that MEHO was more efficient compared with the previous operating condition.

4.2.6. Image Safety Model

Shankar et al. [149] proposed an image safety model based on the EHO algorithm. Two keys, a general public key and a non-public key, were optimized by utilizing adaptive EHO (AEHO). The device was optimized by a hybrid algorithm applying encryption and optimization techniques which mixed the functionality of encryption and digital signatures. The experimental results indicated that the confidentiality of the image was ultimately upheld.

Chibani et al. [150] introduced EHO into the quality of service (QoS) aware web service composition. It was shown that the proposed method offered excellent performances compared with PSO in terms of convergence speed, scalability, and fitness evaluations.

4.2.7. Image Processing

Tuba et al. [151] used the EHO algorithm to solved multilevel image thresholding problems based on Kapur and Otsu's criteria. The proposed algorithm was compared with four swarm intelligence approaches. The experimental results concluded that the EHO algorithm successfully solved multilevel thresholding problems and additionally had smaller variance.

Jino et al. [152] presented the short review of nature-inspired optimization algorithms, such as EHO [105], BAs [91], ACO [55], ABCs [53], PSO [2], FAs [69], bumble bees mating (BBM), and CSO [104]. These algorithms were applied to advanced image processing fields.

Jayanth et al. [153] used the EHO algorithm to classify the high spatial resolution multispectral image classification. According to the fitness function, EHO determines the information of class and multispectral pixels. When compared with the SVM method, the experimental results of two datasets demonstrated that the proposed method improved overall accuracy by 10.7% for dataset 1 and 6.63% for dataset 2.

Cardoso et al. [154] used EHO to improve the search for the maximum correlation point of the image. The search process was implemented in software based on an embedded general purpose processor. The performance results showed that the proposed method outperformed other optimization metaheuristics, which were PSO [2] and ES [79].

4.2.8. Wireless Sensor Networks

Correia et al. [155] applied the EHO algorithm to solve the energy-based source localization problem for wireless sensors networks. The energy decay model between two sensor nodes was matched through key optimized parameters of the EHO algorithm. Comparing the performance between the proposed method and existing non-metaheuristic algorithms, EHO significantly reduced the estimation error in environments with high noise power. In addition, EHO represented an excellent balance between estimation accuracy and computational complexity.

Strumberger et al. [156] solved localization problems for wireless sensor networks using the EHO algorithm. According to the simulation results and comparative analysis with other state-of-the-art algorithms, EHO found the coordinates of unknown nodes randomly deployed in the monitoring field, which proved to be robust and efficient metaheuristics when tackling wireless sensor network localization.

Kaur et al. [157] proposed a novel and energy-efficient approach based on EHO to improve the span of energy in nodes of an underwater network. In the proposed approach, a dynamic cluster head in underwater wireless networks was formed by the behavior of the elephants selecting their heads. It was demonstrated that the EHO algorithm was a promising algorithm for tackling multiple parameters of underwater networks.

4.2.9. Feature Selection

Xu et al. [158] proposed an improved elephant herding optimization (IEHO) algorithm for feature selection in several datasets and distributed environments, which effectively reduced the running time of the algorithm under the premise of ensuring classification accuracy. The experiments showed that the classification efficiency of the IEHO algorithm significantly outperformed other optimization algorithms, such as PSO [2] and EHO [105].

4.2.10. Optimal Power Flow

Dhillon et al. [159] applied EHO to mitigate frequency deviations under sudden variations in demand on the automatic generation control of an interconnected power system. The outcomes of the EHO-based automatic generation control was compared with PSO-based automatic generation control. It was concluded that the settling time of the EHO-based strategy took less time than the PSO-based strategy.

Kuchibhatla et al. [160] used an EHO algorithm to improve the power quality (PQ) and reduce the harmonic distortion in a photovoltaic (PV) interconnected wind energy conversion system (WECS). The performances of three methods (EHO [105], BAs [91], and FAs [69]) were evaluated. The obtained results showed that the proposed method enhanced the performance of the grid-connected hybrid energy system.

Sambariya et al. [161] used EHO to adjust the parameters of a PID controller for the load frequency control of a single-area reheat power system. The solution results showed that the proposed technique obtained better robustness compared with the PID controller.

4.2.11. Distribution Systems

Prasad et al. [162] used EHO to determine the optimal distributed generation (DG) unit size. The proposed model was performed on two types of DG (DG operating at 0.9 power factor lag and DG operating at unity power factor). The numerical results indicated that the EHO algorithm obtained overall better results compared with other algorithms in terms of reducing power consumption.

Vijay et al. [163] applied the EHO technique to the optimal placement and sizing of distributed generation on an electric distribution network. EHO was tested on a 5-bus radial distribution system. The results indicated that the overloading of the equipment, active power, reactive power, and production cost of electricity were reduced, which was more intelligent and precise for the allocation of distributed generation in an electric distribution network.

4.3. Constrained Optimization

4.3.1. Linear and Nonlinear Constrained Optimization

Strumberger et al. [164] presented a hybridized elephant herding optimization (HEHO) algorithm to solve constrained optimization problems. Thirteen standard constrained benchmark functions were conducted for evaluating the efficiency and robustness of the HEHO algorithm. The simulation results

were compared with other state-of-the-art algorithms, such as firefly algorithms, seeker optimization algorithms, and self-adaptive penalty function genetic algorithms. The study results showed that the proposed HEHO was more efficient than the other reported algorithms.

4.3.2. Economic Dispatch Problems

Economic-Based Dispatch Problem

Singh et al. [165] proposed a new modified EHO called MEHO. MEHO was further applied to solve the optimization of linear as well as nonlinear cost functions for economic load dispatch problems. The results obtained showed that the total operating cost obtained by MEHO was less than that of EHO [105], PSO [2], and ACO [55]. The results showed that the MEHO methods had potential for solving linear as well as nonlinear optimization problems.

Stochastic Inequality Constrained Optimization Problems

Horng et al. [166] presented a heuristic method coupling EHO with ordinal optimization (EHOO) to resolve stochastic inequality constrained optimization problems. The proposed method utilized an improved elephant herding optimization to achieve diversification with an accelerated optimal computing budget allocation. The simulation experiment results were obtained by EHOO and compared with three optimization methods (PSO [2], GAs [1], and ES [79]). The results showed that the EHOO approach obtained higher computational efficiency than the other three comparative methods.

4.4. Multi-Objective Optimization

4.4.1. QoS Aware Web Service Composition Optimization

Sadouki et al. [167] proposed a new discrete multi-objective metaheuristic bio-inspired pareto-based approach based on the EHO algorithm to solve the QoS aware web service composition problem. Compared with the multi-objective particle swarm optimization (MOPSO) algorithm and strength pareto evolutionary algorithm 2 (SPEA2), the results showed that the presented method significantly outperformed MOPSO and SPEA2 in terms of set coverage and spacing metrics.

4.4.2. Civil Engineering

Adarsha et al. [168] introduced a hybridized technique named elephant herding optimization-based artificial neural network (EHO-ANN). Furthermore, the complicated experimental procedures for finding the elastic modulus of concrete was solved by the EHO-ANN. The performance of the EHO-ANN algorithm was compared with that of linear regression, empirical formula, and test correlation coefficient (CC). The results showed that the EHO-ANN was more accurate than other methods in predicting the elastic modulus of concrete.

4.4.3. Structural optimization

Jafari et al. [169] combined the advantage of elephant herding optimization (EHO) and the cultural algorithm (CA) and proposed a hybrid algorithm (EHOC). In EHOC, EHO was improved by using the belief space defined by the cultural algorithm. The performance of the EHOC algorithm was evaluated on eight mathematical optimization problems and four truss weight minimization problems. The solution results showed that EHOC was capable of accelerating the convergence rate effectively compared with the CA and EHO.

5. Conclusions and Future Directions

In this paper, tens of research articles related to the EHO algorithm were reviewed. We also discussed the application of the EHO variants in continuous optimization, combinatorial optimization, constrained optimization, and multi-objective optimization. Researchers improved the EHO algorithms

and successfully applied them to various optimization fields. This algorithm has proved to be a promising tool for many optimization problems and engineering applications. However, several aspects of the EHO method that should be further studied, and are as follows:

(1) Most researchers have merely focused on the optimization effects of EHO. There is not sufficient explanation for theoretical analysis. Therefore, strengthening the theoretical analysis of EHO and the mathematical model will remain a challenge in future research.

(2) Employing EHO to solve unsolved optimization problems, especially multi-objective optimization problems, needs to be studied in more depth.

(3) Hybridizing EHO with other algorithm components, such as differential evolution and hill climbing, is another interesting topic for future research [170].

(4) EHO has achieved some notable accomplishments in solving discrete and continuous optimization problems. Therefore, expanding the application scope of EHO and designing suitable optimization operators should be considered in future research.

(5) EHO has a lower level of constrained optimization than similar methods. This is undoubtedly a shortcoming of EHO. Therefore, more research should be carried out to expand EHO for more constrained optimization applications.

Author Contributions: Conceptualization, J.L.; research literature, H.L.; literature search, G.-G.W. and A.H.A.; writing—original draft preparation, J.L.; writing—review and editing, H.L.; funding acquisition, G.-G.W. and A.H.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by an Industry–University Cooperative Education Project (Grant No. 201802046038), Doctoral Foundation of Wuhan Technology and Business University (No. D2019010), and National Natural Science Foundation of China (No. 41576011, No. U1706218, No. 41706010, and No. 61503165).

Acknowledgments: The authors would like to thank the anonymous reviewers and the editor for their careful reviews and constructive suggestions to help us improve the quality of this paper.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Santucci, V.; Baioletti, M.; Milani, A. An algebraic framework for swarm and evolutionary algorithms in combinatorial optimization. *Swarm Evol. Comput.* **2020**, *55*, 100673. [CrossRef]
2. Wang, G.-G.; Tan, Y. Improving metaheuristic algorithms with information feedback models. *IEEE Trans. Cybern.* **2019**, *49*, 542–555. [CrossRef] [PubMed]
3. Valentino, S.; Alfredo, M.; Fabio, C. An optimisation-driven prediction method for automated diagnosis and prognosis. *Mathematics* **2019**, *7*, 1051. [CrossRef]
4. Alfredo, M.; Valentino, S. Asynchronous differential evolution. *IEEE Congr. Evol. Comput.* **2010**, *7*, 18–23.
5. Sang, H.-Y.; Pan, Q.-K.; Duan, P.-Y.; Li, J.-Q. An effective discrete invasive weed optimization algorithm for lot-streaming flowshop scheduling problems. *J. Intell. Manuf.* **2015**, *29*, 1337–1349. [CrossRef]
6. Sang, H.-Y.; Pan, Q.-K.; Li, J.-Q.; Wang, P.; Han, Y.-Y.; Gao, K.-Z.; Duan, P. Effective invasive weed optimization algorithms for distributed assembly permutation flowshop problem with total flowtime criterion. *Swarm Evol. Comput.* **2019**, *44*, 64–73. [CrossRef]
7. Pan, Q.-K.; Sang, H.-Y.; Duan, J.-H.; Gao, L. An improved fruit fly optimization algorithm for continuous function optimization problems. *Knowl.-Based Syst.* **2014**, *62*, 69–83. [CrossRef]
8. Gao, D.; Wang, G.-G.; Pedrycz, W. Solving fuzzy job-shop scheduling problem using de algorithm improved by a selection mechanism. *IEEE Trans. Fuzzy Syst.* **2020**. [CrossRef]
9. Santucci, V.; Baioletti, M.; Milani, A. Algebraic differential evolution algorithm for the permutation flowshop scheduling problem with total flowtime criterion. *IEEE Trans. Evolut. Comput.* **2016**, *20*, 682–694. [CrossRef]
10. Li, M.; Xiao, D.; Zhang, Y.; Nan, H. Reversible data hiding in encrypted images using cross division and additive homomorphism. *Signal Process. Image Commun.* **2015**, *39*, 234–248. [CrossRef]
11. Li, M.; Guo, Y.; Huang, J.; Li, Y. Cryptanalysis of a chaotic image encryption scheme based on permutation-diffusion structure. *Signal Process. Image Commun.* **2018**, *62*, 164–172. [CrossRef]
12. Fan, H.; Li, M.; Liu, D.; Zhang, E. Cryptanalysis of a colour image encryption using chaotic apfm nonlinear adaptive filter. *Signal Process.* **2018**, *143*, 28–41. [CrossRef]

13. Zhang, Y.; Gong, D.; Hu, Y.; Zhang, W. Feature selection algorithm based on bare bones particle swarm optimization. *Neurocomputing* **2015**, *148*, 150–157. [CrossRef]
14. Zhang, Y.; Song, X.-F.; Gong, D.-W. A return-cost-based binary firefly algorithm for feature selection. *Inf. Sci.* **2017**, *418–419*, 561–574. [CrossRef]
15. Mao, W.; He, J.; Tang, J.; Li, Y. Predicting remaining useful life of rolling bearings based on deep feature representation and long short-term memory neural network. *Adv. Mech. Eng.* **2018**, *10*. [CrossRef]
16. Jian, M.; Lam, K.-M.; Dong, J. Facial-feature detection and localization based on a hierarchical scheme. *Inf. Sci.* **2014**, *262*, 1–14. [CrossRef]
17. Fan, L.; Xu, S.; Liu, D.; Ru, Y. Semi-supervised community detection based on distance dynamics. *IEEE Access* **2018**, *6*, 37261–37271. [CrossRef]
18. Wang, G.-G.; Chu, H.E.; Mirjalili, S. Three-dimensional path planning for ucav using an improved bat algorithm. *Aerosp. Sci. Technol.* **2016**, *49*, 231–238. [CrossRef]
19. Wang, G.; Guo, L.; Duan, H.; Liu, L.; Wang, H.; Shao, M. Path planning for uninhabited combat aerial vehicle using hybrid meta-heuristic de/bbo algorithm. *Adv. Sci. Eng. Med.* **2012**, *4*, 550–564. [CrossRef]
20. Wang, G.-G.; Cai, X.; Cui, Z.; Min, G.; Chen, J. High performance computing for cyber physical social systems by using evolutionary multi-objective optimization algorithm. *IEEE Trans. Emerg. Top. Comput.* **2017**. [CrossRef]
21. Cui, Z.; Sun, B.; Wang, G.-G.; Xue, Y.; Chen, J. A novel oriented cuckoo search algorithm to improve dv-hop performance for cyber-physical systems. *J. Parallel Distrib. Comput.* **2017**, *103*, 42–52. [CrossRef]
22. Jian, M.; Lam, K.-M.; Dong, J. Illumination-insensitive texture discrimination based on illumination compensation and enhancement. *Inf. Sci.* **2014**, *269*, 60–72. [CrossRef]
23. Wang, G.-G.; Guo, L.; Duan, H.; Liu, L.; Wang, H. The model and algorithm for the target threat assessment based on elman_adaboost strong predictor. *Acta Electron. Sin.* **2012**, *40*, 901–906.
24. Jian, M.; Lam, K.M.; Dong, J.; Shen, L. Visual-patch-attention-aware saliency detection. *IEEE Trans. Cybern.* **2015**, *45*, 1575–1586. [CrossRef] [PubMed]
25. Wang, G.-G.; Lu, M.; Dong, Y.-Q.; Zhao, X.-J. Self-adaptive extreme learning machine. *Neural Comput. Appl.* **2016**, *27*, 291–303. [CrossRef]
26. Li, J.; Li, Y.-X.; Tian, S.-S.; Xia, J.-L. An improved cuckoo search algorithm with self-adaptive knowledge learning. *Neural Comput. Appl.* **2019**. [CrossRef]
27. Liu, G.; Zou, J. Level set evolution with sparsity constraint for object extraction. *IET Image Process.* **2018**, *12*, 1413–1422. [CrossRef]
28. Liu, K.; Gong, D.; Meng, F.; Chen, H.; Wang, G.-G. Gesture segmentation based on a two-phase estimation of distribution algorithm. *Inf. Sci.* **2017**, *394–395*, 88–105. [CrossRef]
29. Rizk-Allah, R.M.; El-Sehiemy, R.A.; Wang, G.-G. A novel parallel hurricane optimization algorithm for secure emission/economic load dispatch solution. *Appl. Soft Comput.* **2018**, *63*, 206–222. [CrossRef]
30. Rizk-Allah, R.M.; El-Sehiemy, R.A.; Deb, S.; Wang, G.-G. A novel fruit fly framework for multi-objective shape design of tubular linear synchronous motor. *J. Supercomput.* **2017**, *73*, 1235–1256. [CrossRef]
31. Yi, J.-H.; Deb, S.; Dong, J.; Alavi, A.H.; Wang, G.-G. An improved nsga-iii algorithm with adaptive mutation operator for big data optimization problems. *Future Gener. Comput. Syst.* **2018**, *88*, 571–585. [CrossRef]
32. Liu, G.; Deng, M. Parametric active contour based on sparse decomposition for multi-objects extraction. *Signal Process.* **2018**, *148*, 314–321. [CrossRef]
33. Sun, J.; Miao, Z.; Gong, D.; Zeng, X.-J.; Li, J.; Wang, G.-G. Interval multi-objective optimization with memetic algorithms. *IEEE Trans. Cybern.* **2019**. [CrossRef]
34. Zhang, Y.; Wang, G.-G.; Li, K.; Yeh, W.-C.; Jian, M.; Dong, J. Enhancing moea/d with information feedback models for large-scale many-objective optimization. *Inf. Sci.* **2020**, *522*, 1–16. [CrossRef]
35. Gu, Z.-M.; Wang, G.-G. Improving NSGA-III algorithms with information feedback models for large-scale many-objective optimization. *Future Gener. Comput. Syst.* **2020**, *107*, 49–69. [CrossRef]
36. Srikanth, K.; Panwar, L.K.; Panigrahi, B.K.; Herrera-Viedma, E.; Sangaiah, A.K.; Wang, G.-G. Meta-heuristic framework: Quantum inspired binary grey wolf optimizer for unit commitment problem. *Comput. Electr. Eng.* **2018**, *70*, 243–260. [CrossRef]
37. Li, J.; Xiao, D.-D.; Lei, H.; Zhang, T.; Tian, T. Using cuckoo search algorithm with q-learning and genetic operation to solve the problem of logistics distribution center location. *Mathematics (Basel)* **2020**, *8*, 149. [CrossRef]

38. Chen, S.; Chen, R.; Wang, G.-G.; Gao, J.; Sangaiah, A.K. An adaptive large neighborhood search heuristic for dynamic vehicle routing problems. *Comput. Electr. Eng.* **2018**. [CrossRef]
39. Feng, Y.; Wang, G.-G. Binary moth search algorithm for discounted {0–1} knapsack problem. *IEEE Access* **2018**, *6*, 10708–10719. [CrossRef]
40. Feng, Y.; Wang, G.-G.; Wang, L. Solving randomized time-varying knapsack problems by a novel global firefly algorithm. *Eng. Comput.* **2018**, *34*, 621–635. [CrossRef]
41. Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning*; Addison-Wesley: New York, NY, USA, 1998.
42. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; IEEE: Perth, Australia, 1995; pp. 1942–1948.
43. Wang, G.-G.; Gandomi, A.H.; Yang, X.-S.; Alavi, A.H. A novel improved accelerated particle swarm optimization algorithm for global numerical optimization. *Eng. Comput.* **2014**, *31*, 1198–1220. [CrossRef]
44. Sun, J.; Feng, B.; Xu, W. Particle Swarm Optimization with Particles Having Quantum Behavior. In Proceedings of the Congress on Evolutionary Computation (CEC 2004), Portland, OR, USA, 19–23 June 2004; IEEE: Portland, OR, USA, 2004; pp. 325–331.
45. Adewumi, A.O.; Arasomwan, M.A. On the performance of particle swarm optimisation with(out) some control parameters for global optimisation. *Int. J. Bio-Inspir. Com.* **2016**, *8*, 14–32. [CrossRef]
46. Storn, R.; Price, K. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
47. Xu, Z.; Unveren, A.; Acan, A. Probability collectives hybridised with differential evolution for global optimisation. *Int. J. Bio-Inspir. Com.* **2016**, *8*, 133–153. [CrossRef]
48. Wang, G.-G.; Zhao, X.; Deb, S. A Novel Monarch Butterfly Optimization with Greedy Strategy and Self-Adaptive Crossover Operator. In Proceedings of the 2015 IEEE 2nd Intl, Conference on Soft Computing & Machine Intelligence (ISCMI 2015), Hong Kong, China, 23–24 November 2015; pp. 45–50.
49. Wang, G.-G.; Deb, S.; Zhao, X.; Cui, Z. A new monarch butterfly optimization with an improved crossover operator. *Oper. Res. Int. J.* **2018**, *18*, 731–755. [CrossRef]
50. Feng, Y.; Wang, G.-G.; Li, W.; Li, N. Multi-strategy monarch butterfly optimization algorithm for discounted {0–1} knapsack problem. *Neural Comput. Appl.* **2018**, *30*, 3019–3036. [CrossRef]
51. Wang, G.-G.; Deb, S.; Cui, Z. Monarch butterfly optimization. *Neural Comput. Appl.* **2019**, *31*, 1995–2014. [CrossRef]
52. Feng, Y.; Wang, G.-G.; Deb, S.; Lu, M.; Zhao, X. Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization. *Neural Comput. Appl.* **2017**, *28*, 1619–1634. [CrossRef]
53. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
54. Wang, G.-G.; Deb, S.; Coelho, L.D.S. Earthworm optimization algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Int. J. Bio.-Inspir. Com.* **2018**, *12*, 1–22. [CrossRef]
55. Dorigo, M.; Stutzle, T. *Ant Colony Optimization*; MIT Press: Cambridge, UK, 2004.
56. Li, J.; Xiao, D.-D.; Zhang, T.; Liu, C.; Li, Y.-X. Multi-swarm cuckoo search algorithm with q-learning model. *Comput. J.* **2020**. [CrossRef]
57. Yang, X.-S.; Deb, S. Cuckoo search via lévy flights. In Proceedings of the World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), Coimbatore, India, 9–11 December 2009; Abraham, A., Carvalho, A., Herrera, F., Pai, V., Eds.; IEEE Publications: Coimbatore, India, 2009; pp. 210–214.
58. Li, X.; Wang, J.; Yin, M. Enhancing the performance of cuckoo search algorithm using orthogonal learning method. *Neural Comput. Appl.* **2013**, *24*, 1233–1247. [CrossRef]
59. Li, X.; Yin, M. Modified cuckoo search algorithm with self adaptive parameter method. *Inf. Sci.* **2015**, *298*, 80–97. [CrossRef]
60. Wang, G.-G.; Gandomi, A.H.; Zhao, X.; Chu, H.E. Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. *Soft Comput.* **2016**, *20*, 273–285. [CrossRef]
61. Wang, G.-G.; Deb, S.; Gandomi, A.H.; Zhang, Z.; Alavi, A.H. Chaotic cuckoo search. *Soft Comput.* **2016**, *20*, 3349–3362. [CrossRef]
62. Wang, G.; Guo, L.; Duan, H.; Liu, L.; Wang, H.; Wang, J. A hybrid meta-heuristic de/cs algorithm for ucav path planning. *J. Inform. Comput. Sci.* **2012**, *9*, 4811–4818.

63. Gandomi, A.H.; Alavi, A.H. Krill herd: A new bio-inspired optimization algorithm. *Commun. Nonlinear Sci.* **2012**, *17*, 4831–4845. [CrossRef]
64. Li, Z.-Y.; Yi, J.-H.; Wang, G.-G. A new swarm intelligence approach for clustering based on krill herd with elitism strategy. *Algorithms* **2015**, *8*, 951–964. [CrossRef]
65. Wang, G.-G.; Gandomi, A.H.; Alavi, A.H.; Deb, S. A multi-stage krill herd algorithm for global numerical optimization. *Int. J. Artif. Intell. Tools* **2016**, *25*, 1550030. [CrossRef]
66. Wang, G.-G.; Gandomi, A.H.; Alavi, A.H. An effective krill herd algorithm with migration operator in biogeography-based optimization. *Appl. Math. Model* **2014**, *38*, 2454–2462. [CrossRef]
67. Wang, G.-G.; Gandomi, A.H.; Alavi, A.H.; Gong, D. A comprehensive review of krill herd algorithm: Variants, hybrids and applications. *Artif. Intell. Rev.* **2019**, *51*, 119–148. [CrossRef]
68. Gandomi, A.H.; Yang, X.-S.; Alavi, A.H. Mixed variable structural optimization using firefly algorithm. *Comput. Struct.* **2011**, *89*, 2325–2336. [CrossRef]
69. Yang, X.S. Firefly algorithm, stochastic test functions and design optimisation. *Int. J. Bio-Inspir. Com.* **2010**, *2*, 78–84. [CrossRef]
70. Wang, G.-G.; Guo, L.; Duan, H.; Wang, H. A new improved firefly algorithm for global numerical optimization. *J. Comput. Theor. Nanosci.* **2014**, *11*, 477–485. [CrossRef]
71. Gálvez, A.; Iglesias, A. New memetic self-adaptive firefly algorithm for continuous optimisation. *Int. J. Bio-Inspir. Com.* **2016**, *8*, 300–317. [CrossRef]
72. Nasiri, B.; Meybodi, M.R. History-driven firefly algorithm for optimisation in dynamic and uncertain environments. *Int. J. Bio.-Inspir. Com.* **2016**, *8*, 326–339. [CrossRef]
73. Wang, G.; Guo, L.; Duan, H.; Liu, L.; Wang, H. A modified firefly algorithm for ucav path planning. *Int. J. Hybrid Inf. Technol.* **2012**, *5*, 123–144.
74. Kirkpatrick, S.; Gelatt, C.D., Jr.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [CrossRef]
75. Shah-Hosseini, H. The intelligent water drops algorithm: A nature-inspired swarm-based optimization algorithm. *Int. J. Bio-Inspir. Com.* **2009**, *1*, 71–79. [CrossRef]
76. Eskandar, H.; Sadollah, A.; Bahreininejad, A.; Hamdi, M. Water cycle algorithm—A novel metaheuristic optimization method for solving constrained engineering optimization problems. *Comput. Struct.* **2012**, *110–111*, 151–166. [CrossRef]
77. Wang, G.-G. Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Memetic Comput.* **2018**, *10*, 151–164. [CrossRef]
78. Zhao, R.; Tang, W. Monkey algorithm for global numerical optimization. *J. Uncertain Syst.* **2008**, *2*, 165–176.
79. Beyer, H. *The Theory of Evolution Strategies*; Springer: New York, NY, USA, 2001.
80. Penev, K.; Littlefair, G. Free search-a comparative analysis. *Inf. Sci.* **2005**, *172*, 173–193. [CrossRef]
81. Baluja, S. *Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning*; CMU-CS-94-163; Carnegie Mellon University: Pittsburgh, PA, USA, 1994.
82. Simon, D. Biogeography-based optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [CrossRef]
83. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Let a biogeography-based optimizer train your multi-layer perceptron. *Inf. Sci.* **2014**, *269*, 188–209. [CrossRef]
84. Duan, H.; Zhao, W.; Wang, G.; Feng, X. Test-sheet composition using analytic hierarchy process and hybrid metaheuristic algorithm ts/bbo. *Math. Probl. Eng.* **2012**, *2012*, 1–22. [CrossRef]
85. Wang, G.; Guo, L.; Duan, H.; Liu, L.; Wang, H. Dynamic deployment of wireless sensor networks by biogeography based optimization algorithm. *J. Sens. Actuator Netw.* **2012**, *1*, 86–96. [CrossRef]
86. Mirjalili, S. Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Comput. Appl.* **2016**, *27*, 1053–1073. [CrossRef]
87. Gandomi, A.H. Interior search algorithm (ISA): A novel approach for global optimization. *ISA Trans.* **2014**, *53*, 1168–1183. [CrossRef]
88. Shi, Y. An optimization algorithm based on brainstorming process. *Int. J. Swarm Intell. Res.* **2011**, *2*, 35–62. [CrossRef]
89. Shi, Y.; Xue, J.; Wu, Y. Multi-objective optimization based on brain storm optimization algorithm. *Int. J. Swarm Intell. Res.* **2013**, *4*, 1–21. [CrossRef]

90. Gandomi, A.H.; Yang, X.-S.; Alavi, A.H.; Talatahari, S. Bat algorithm for constrained optimization tasks. *Neural Comput. Appl.* **2013**, *22*, 1239–1255. [CrossRef]
91. Yang, X.S.; Gandomi, A.H. Bat algorithm: A novel approach for global engineering optimization. *Eng. Comput.* **2012**, *29*, 464–483. [CrossRef]
92. Mirjalili, S.; Mirjalili, S.M.; Yang, X.-S. Binary bat algorithm. *Neural Comput. Appl.* **2013**, *25*, 663–681. [CrossRef]
93. Cai, X.; Gao, X.-Z.; Xue, Y. Improved bat algorithm with optimal forage strategy and random disturbance strategy. *Int. J. Bio-Inspir. Com.* **2016**, *8*, 205–214. [CrossRef]
94. Wang, G.; Guo, L. A novel hybrid bat algorithm with harmony search for global numerical optimization. *J. Appl. Math.* **2013**, *2013*, 1–21. [CrossRef]
95. Wang, G.-G.; Chang, B.; Zhang, Z. A Multi-Swarm Bat Algorithm for Global Optimization. In Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC 2015), Sendai, Japan, 25–28 May 2015; IEEE: Sendai, Japan, 2015; pp. 480–485.
96. Wang, G.-G.; Lu, M.; Zhao, X.-J. An improved bat algorithm with variable neighborhood search for global optimization. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (IEEE CEC 2016), Vancouver, BC, Canada, 24–29 July 2016; pp. 1773–1778.
97. Yang, X.-S. *Nature-Inspired Metaheuristic Algorithms*, 2nd ed.; Luniver Press: Frome, UK, 2010.
98. Khatib, W.; Fleming, P. The stud ga: A mini revolution? In *Parallel Problem Solving from Nature-ppsn v*; Eiben, A., Bäck, T., Schoenauer, M., Schwefel, H.-P., Eds.; Springer: Berlin/Heidelberg, Germany; London, UK, 1998; Volume 1498, pp. 683–691.
99. Geem, Z.W.; Kim, J.H.; Loganathan, G.V. A new heuristic optimization algorithm: Harmony search. *Simulation* **2001**, *76*, 60–68. [CrossRef]
100. Wang, G.; Guo, L.; Duan, H.; Wang, H.; Liu, L.; Shao, M. Hybridizing harmony search with biogeography based optimization for global numerical optimization. *J. Comput. Theor. Nanosci.* **2013**, *10*, 2318–2328. [CrossRef]
101. Niknam, T.; Fard, A.K. Optimal energy management of smart renewable micro-grids in the reconfigurable systems using adaptive harmony search algorithm. *Int. J. Bio-Inspir. Com.* **2016**, *8*, 184–194. [CrossRef]
102. Rezoug, A.; Boughaci, D. A self-adaptive harmony search combined with a stochastic local search for the 0–1 multidimensional knapsack problem. *Int. J. Bio-Inspir. Com.* **2016**, *8*, 234–239. [CrossRef]
103. Tan, Y. *Fireworks Algorithm-A Novel Swarm Intelligence Optimization Method*; Springer: Berlin/Heidelberg, Germany, 2015.
104. Meng, X.; Liu, Y.; Gao, X.; Zhang, H. A new bio-inspired algorithm: Chicken swarm optimization. In Proceedings of the Advances in Swarm Intelligence (ICSI 2014), Hefei, China, 17–20 October 2014; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8794, pp. 86–94.
105. Wang, G.-G.; Deb, S.; Coelho, L.d.S. Elephant Herding Optimization. In Proceedings of the 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI 2015), Bali, Indonesia, 7–9 December 2015; IEEE: Bali, Indonesia, 2015; pp. 1–5.
106. Tuba, E.; Capor-Hrosik, R.; Alihodzic, A.; Jovanovic, R.; Tuba, M. Chaotic Elephant Herding Optimization Algorithm. In Proceedings of the 2018 IEEE 16th World Symposium on Applied Machine Intelligence and Informatics (SAMI 2018), Kosice and Herlany, Slovakia, 7–10 February 2018; IEEE: Kosice and Herlany, Slovakia, 2018; pp. 213–216.
107. Li, J.; Guo, L.; Li, Y.; Liu, C. Enhancing elephant herding optimization with novel individual updating strategies for large-scale optimization problems. *Mathematics* **2019**, *7*, 395. [CrossRef]
108. Xu, H.; Cao, Q.; Fang, C.; Fu, Y.; Su, J.; Wei, S.; Bykovyy, P. Application of Elephant Herd Optimization Algorithm Based on Levy Flight Strategy in Intrusion Detection. In Proceedings of the 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), Lviv, Ukraine, 20–21 September 2018; IEEE: Lviv, Ukraine, 2018; pp. 16–20.
109. Xu, H.; Cao, Q.; Fu, H.; Fu, C.; Chen, H.; Su, J. *Application of Support Vector Machine Model Based on an Improved Elephant Herding Optimization Algorithm in Network Intrusion Detection*; Springer: Singapore, 2019; pp. 283–295.
110. Hakli, H. Elephant herding optimization using multi-search strategy for continuous optimization problems. *Acad. Platf. J. Eng. Sci.* **2019**, *7*, 261–268. [CrossRef]

111. Tuba, E.; Dolicanin-Djekic, D.; Jovanovic, R.; Simian, D.; Tuba, M. *Combined Elephant Herding Optimization Algorithm with k-Means for Data Clustering*; Springer: Singapore, 2019; pp. 665–673.
112. Chakraborty, F.; Roy, P.K.; Nandi, D. Oppositional elephant herding optimization with dynamic cauchy mutation for multilevel image thresholding. *Evol. Intell.* **2019**, *12*, 445–467. [CrossRef]
113. Chowdary, K.U.; Prabhakara Rao, B. Performance improvement in mimo-ofdm systems based on adaptive whale elephant herd optimization algorithm. *Int. J. Eng. Adv. Technol.* **2019**, *9*, 6651–6657.
114. Rashwan, Y.I.; Elhosseini, M.A.; El Sehiemy, R.A.; Gao, X.Z. On the performance improvement of elephant herding optimization algorithm. *Knowl.-Based Syst.* **2019**, *166*, 58–70.
115. ElShaarawy, I.A.; Houssein, E.H.; Ismail, F.H.; Hassanien, A.E. An exploration-enhanced elephant herding optimization. *Eng Comput.* **2019**, *36*, 3029–3046. [CrossRef]
116. Ismaeel, A.A.K.; Elshaarawy, I.A.; Houssein, E.H.; Ismail, F.H.; Hassanien, A.E. Enhanced elephant herding optimization for global optimization. *IEEE Access* **2019**, *7*, 34738–34752. [CrossRef]
117. Veera manikandan, P.; Selvaperumal, S. A fuzzy-elephant herding optimization technique for maximum power point tracking in the hybrid wind-solar system. *Int. Trans. Electr. Energy Syst.* **2019**. [CrossRef]
118. Arora, P.; Dixit, A. The hybrid optimization algorithm for load balancing in cloud. *Int. J. Eng. Adv. Technol.* **2019**, *8*, 67–71.
119. Bukhsh, R.; Javaid, N.; Iqbal, Z.; Ahmed, U.; Ahmad, Z.; Iqbal, M.N. Appliances Scheduling Using Hybrid Scheme of Genetic Algorithm and Elephant Herd Optimization for Residential Demand Response. In Proceedings of the 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA 2018), Krakow, Poland, 16–18 May 2018; IEEE: Krakow, Poland, 2018; pp. 210–217.
120. Strumberger, I.; Minovic, M.; Tuba, M.; Bacanin, N. Performance of elephant herding optimization and tree growth algorithm adapted for node localization in wireless sensor networks. *Sensors* **2019**, *19*, 2515. [CrossRef]
121. Satapathy, P.; Pradhan, S.K.; Hota, S. Development of a novel neural network model for brain image classification. *Int. J. Recent Technol. Eng.* **2019**, *8*, 7230–7235.
122. Hakli, H. A novel approach based on elephant herding optimization for constrained optimization problems. *Selçuk Üniversitesi Mühendislik Bilim ve Teknoloji Dergisi* **2019**, *7*, 405–419. [CrossRef]
123. Hakli, H. Bineho: A new binary variant based on elephant herding optimization algorithm. *Neural Comput. Appl.* **2020**. [CrossRef]
124. Jaiprakash, K.P.; Nanda, S.J. Classifying Physical Actions of Human Models Using Multi-Objective Clustering Based on Elephant Herding Algorithm. In Proceedings of the 1st International Conference on Pervasive Computing Advances and Applications (PerCAA 2019), Jaipur, India, 8–10 January 2019; Bundele, M., Dey, N., Madria, S.K., Eds.; Elsevier B.V.: Jaipur, India, 2019; pp. 84–91.
125. Meena, N.K.; Parashar, S.; Swarnkar, A.; Gupta, N.; Niazi, K.R. Improved elephant herding optimization for multiobjective der accommodation in distribution systems. *IEEE Trans. Ind. Inform.* **2018**, *14*, 1029–1039. [CrossRef]
126. Moayedi, H.; Mu'azu, M.A.; Foong, L.K. Novel swarm-based approach for predicting the cooling load of residential buildings based on social behavior of elephant herds. *Energy Build.* **2020**, *206*, 109579. [CrossRef]
127. Kowsalya, S.; Periasamy, P.S. Recognition of tamil handwritten character using modified neural network with aid of elephant herding optimization. *Multimed Tools Appl.* **2019**, *78*, 25043–25061. [CrossRef]
128. Sahlol, A.T.; Ismail, F.H.; Abdeldaim, A.; Hassanien, A.E. Elephant Herd Optimization with Neural Networks: A Case Study on Acute Lymphoblastic Leukemia Diagnosis. In Proceedings of the 2017 12th International Conference on Computer Engineering and Systems (ICCES 2017), Cairo, Egypt, 19–20 December 2017; IEEE: Cairo, Egypt, 2017; pp. 657–662.
129. Kaur, S. Energy optimization for underwater sensor network using nature inspired technique. *Int. J. Innov. Technol. Explor. Eng.* **2019**, *8*, 161–164.
130. Alihodzic, A.; Tuba, E.; Capor-Hrosik, R.; Dolicanin, E.; Tuba, M. Unmanned Aerial Vehicle Path Planning Problem by Adjusted Elephant Herding Optimization. In Proceedings of the 2017 25th Telecommunication Forum (TELFOR), Belgrade, Serbia, 21–22 November 2017; IEEE: Belgrade, Serbia, 2017; pp. 1–4.
131. Rani, R.R.; Ramyachitra, D.; Brindhadevi, A. Detection of dynamic protein complexes through markov clustering based on elephant herd optimization approach. *Sci. Rep.* **2019**, *9*, 11106. [CrossRef]
132. Jaiprakash, K.P.; Nanda, S.J. *Elephant Herding Algorithm for Clustering*; Springer: Singapore, 2019; pp. 317–325.

133. Hassanien, A.E.; Kilany, M.; Houssein, E.H.; AlQaheri, H. Intelligent human emotion recognition based on elephant herding optimization tuned support vector regression. *Biomed. Signal Process. Control* **2018**, *45*, 182–191. [CrossRef]
134. Hassanien, A.E.; Kilany, M.; Houssein, E.H. Combining support vector machine and elephant herding optimization for cardiac arrhythmias. *arXiv* **2018**, arXiv:1806.08242.
135. Tuba, E.; Stanimirovic, Z. Elephant Herding Optimization Algorithm for Support Vector Machine Parameters Tuning. In Proceedings of the IEEE International Conference on Electronics, Computers and Artificial Intelligence (ECAI 2017), Targoviste, Romania, 29 June–1 July 2017; pp. 1–4.
136. Tuba, E.; Ribic, I.; Capor-Hrosik, R.; Tuba, M. Support vector machine optimized by elephant herding algorithm for erythemato-squamous diseases detection. *Procedia Comput. Sci.* **2017**, *122*, 916–923. [CrossRef]
137. Sambariya, D.K.; Fagna, R. A Novel Elephant Herding Optimization Based pid Controller Design for Load Frequency Control in Power System. In Proceedings of the 2017 International Conference on Computer, Communications and Electronics (Comptelix), Jaipur, India, 1–2 July 2017; IEEE: Jaipur, India, 2017; pp. 595–600.
138. Almufti, S.; Boya Marqas, R.; Asaad, R.R. Comparative study between elephant herding optimization (eho) and u-turning ant colony optimization (u-taco) in solving symmetric traveling salesman problem (stsp). *J. Adv. Comput. Sci. Technol.* **2019**, *8*. [CrossRef]
139. Darmawan, H.; Rini, D.P.; Arsalan, O. Penerapan Algoritma Elephant Herding Optimization pada Permasalahan Knapsack 0-1. Undergraduate thesis, Sriwijaya University, Kota Palembang, Sumatera Selatan, 2019. Undergraduate Thesis, Sriwijaya University, Kota Palembang, Sumatera Selatan, 2019.
140. Correia, S.D.; Beko, M.; Cruz, L.A.D.S.; Tomic, S. Implementation and Validation of Elephant Herding Optimization Algorithm for Acoustic Localization. In Proceedings of the 2018 26th Telecommunications Forum (TELFOR), Belgrade, Serbia, 20–21 November 2018; IEEE: Belgrade, Serbia, 2018; pp. 1–4.
141. Parashar, S.; Swarnkar, A.; Niazi, K.R.; Gupta, N. Stochastic operational management of grid-connected microgrid under uncertainty of renewable resources and load demand. In *Lecture Notes in Electrical Engineering*; Springer: Singapore, 2020; Volume 607, pp. 573–581.
142. Cahig, C.; Villanueva, J.J.; Bersano, R.; Pacis, M. Optimal Virtual Power Plant Scheduling Using Elephant Herding Optimization. In Proceedings of the 10th IEEE International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management, HNICEM 2018, Baguio City, Philippines, 29 November–2 December 2018; IEEE: Baguio City, Philippines, 2019.
143. Sarwar, M.A.; Amin, B.; Ayub, N.; Faraz, S.H.; Khan, S.U.R.; Javaid, N. Scheduling of appliances in home energy management system using elephant herding optimization and enhanced differential evolution. In *Advances in Intelligent Networking and Collaborative Systems, Proceedings of the 9th International Conference on Intelligent Networking and Collaborative Systems (INCoS 2017), Ryerson Univ, Toronto, ON, Canada, 24–26 August 2017*; Springer: Berlin, Germany, 2017; pp. 132–142.
144. Parvez, K.; Aslam, S.; Saba, A.; Aimal, S.; Amjad, Z.; Asif, S.; Javaid, N. Scheduling of appliances in hems using elephant herding optimization and harmony search algorithm. In *Advances on Broad-Band Wireless Computing, Communication and Applications, Proceedings of the 12th International Conference on Broad-Band Wireless Computing, Communication and Applications (BWCCA 2017), Barcelona, Spain, 8–10 November 2017*; Springer: Berlin, Germany, 2017; pp. 62–72.
145. Mohsin, S.M.; Javaid, N.; Madani, S.A.; Akber, S.M.A.; Manzoor, S.; Ahmad, J. Implementing Elephant Herding Optimization Algorithm with Different Operation Time Intervals for Appliance Scheduling in Smart Grid. In Proceedings of the 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA), Krakow, Poland, 16–18 May 2018; IEEE: Krakow, Poland, 2018; pp. 240–249.
146. Gholami, H.R.; Mehdizadeh, E.; Naderi, B. Mathematical models and an elephant herding optimization for multiprocessor-task flexible flow shop scheduling problems in the manufacturing resource planning (mrpii) system. *Scientia Iranica* **2018**. [CrossRef]
147. Fatima, I.; Asif, S.; Shafiq, S.; Fatima, I.; Rahim, M.H.; Javaid, N. Efficient Demand Side Management Using Hybridization of Elephant Herding Optimization and Firefly Optimization. In Proceedings of the 2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA), Krakow, Poland, 16–18 May 2018; IEEE: Krakow, Poland, 2018; pp. 839–845.

148. Luangpaiboon, P. Variable tuning for electrostatic powder coating process via elephant herding optimisation algorithm on modified simplex method. *Int. J. Mech. Eng. Robot. Res.* **2019**, *8*, 807–812. [CrossRef]
149. Shankar, K.; Elhoseny, M.; Perumal, E.; Ilayaraja, M.; Sathesh Kumar, K. An efficient image encryption scheme based on signcryption technique with adaptive elephant herding optimization. In *Cybersecurity and Secure Information Systems: Challenges and Solutions in Smart Environments*; Springer: Cham, Switzerland, 20 June 2019; pp. 31–42.
150. Chibani, S.S.; Tari, A. Elephant herding optimization for service selection in qos-aware web service composition. *Int. J. Comput. Electr. Autom. Control Inf. Eng.* **2017**, *11*, 1045–1049.
151. Tuba, E.; Alihodzic, A.; Tuba, M. Multilevel Image Thresholding Using Elephant Herding Optimization Algorithm. In Proceedings of the 2017 14th International Conference on Engineering of Modern Electric Systems (EMES), Oradea, Romania, 1–2 June 2017; IEEE: Oradea, Romania, 2017; pp. 240–243.
152. Jino Ramson, S.R.; Lova Raju, K.; Vishnu, S.; Anagnostopoulos, T. Nature inspired optimization techniques for image processing-a short review. In *Nature Inspired Optimization Techniques for Image Processing Applications*; Springer: Cham, Switzerland, 20 September 2018; Volume 150, pp. 113–145.
153. Jayanth, J.; Shalini, V.S.; Ashok Kumar, T.; Koliwad, S. Land-use/land-cover classification using elephant herding algorithm. *J. Indian Soc. Remote* **2019**, *47*, 223–232. [CrossRef]
154. De Vasconcelos Cardoso, A.; Nedjah, N.; De Macedo Mourelle, L.; Tavares, Y.M. Co-Design System for Template Matching Using Dedicated co-Processor and Modified Elephant Herding Optimization. In Proceedings of the 2018 IEEE 9th Latin American Symposium on Circuits and Systems (LASCAS 2018), Puerto Vallarta, Mexico, 25–28 February 2018; pp. 1–4.
155. Correia, S.; Beko, M.; Cruz, L.; Tomic, S. Elephant herding optimization for energy-based localization. *Sensors* **2018**, *18*, 2849. [CrossRef]
156. Strumberger, I.; Beko, M.; Tuba, M.; Minovic, M.; Bacanin, N. Elephant Herding Optimization Algorithm for Wireless Sensor Network Localization Problem. In Proceedings of the Technological Innovation for Resilient Systems: 9th IFIP WG 5.5/SOCOLNET Advanced Doctoral Conference on Computing, Electrical and Industrial Systems (DoCEIS 2018), Costa de Caparica, Portugal, 2–4 May 2018; pp. 175–184.
157. Kaur, K.; Randhawa, R. Energy efficient approach for underwater sensor network using elephant herd optimization. *Res. Cell Int. J. Eng. Sci.* **2018**, *30*, 148–160.
158. Xu, H.; Cao, Q.; Fu, H.; Chen, H. Applying an improved elephant herding optimization algorithm with spark-based parallelization to feature selection for intrusion detection. *Int. J. Perform. Eng.* **2019**, *15*, 1600–1610. [CrossRef]
159. Dhillon, S.S.; Agarwal, S.; Wang, G.-G.; Lather, J.S. Automatic generation control of interconnected power systems using elephant herding optimization. In *Lecture Notes in Electrical Engineering*; Springer: Singapore, 2020; pp. 9–18.
160. Kuchibhatla, S.M.; Padmavathi, D.; Rao, R.S. An elephant herding optimization algorithm-based static switched filter compensation scheme for power quality improvement in smart grid. *J. Circuits Syst. Comput.* **2019**. [CrossRef]
161. Sambariya, D.K.; Fagna, R. A Robust Pid Controller for Load Frequency Control of Single Area re-heat Thermal Power Plant Using Elephant Herding Optimization Techniques. In Proceedings of the 2017 IEEE International Conference on Information, Communication, Instrumentation and Control, ICICIC 2017, Indore, India, 17–19 August 2017; pp. 1–6.
162. Prasad, C.H.; Subbaramaiah, K.; Sujatha, P. Cost–benefit analysis for optimal dg placement in distribution systems by using elephant herding optimization algorithm. *Renew. Wind Water Sol.* **2019**, *6*. [CrossRef]
163. Vijay, R.; Abhilash, M. Elephant herding optimization for optimum allocation of electrical distributed generation on distributed power networks. *Asian J. Electr. Sci.* **2018**, *7*, 70–76.
164. Strumberger, I.; Bacanin, N.; Tuba, M. Hybridized Elephant Herding Optimization Algorithm for Constrained Optimization. In Proceedings of the 17th International Conference on Hybrid Intelligent Systems (HIS 2017), Delhi, India, 14–16 December 2017; Springer International Publishing: Delhi, India, 2017; pp. 158–166.
165. Singh, N.; Kumar, M.P.; Kumar, B.S. Effect of valve loading on the thermal power economic load dispatch using new elephant herding optimization. *Int. J. Recent Technol. Eng.* **2019**, *7*, 345–349.
166. Horng, S.-C.; Lin, S.-S. Coupling elephant herding with ordinal optimization for solving the stochastic inequality constrained optimization problems. *Appl. Sci.* **2020**, *10*, 2075. [CrossRef]

167. Sadouki, S.C.; Tari, A. Multi-objective and discrete elephants herding optimization algorithm for qos aware web service composition. *RAIRO-Oper. Res.* **2019**, *53*, 445–459. [CrossRef]
168. Adarsha, B.S.; Harish, N.; Janardhan, P.; Mandal, S. *Elephant Herding Optimization Based Neural Network to Predict Elastic Modulus of Concrete*; Soft Computing for Problem Solving; Das, K.N., Bansal, J.C., Deep, K., Nagar, A.K., Pathipooranam, P., Naidu, R.C., Eds.; Springer: Singapore, 2020; pp. 353–364.
169. Jafari, M.; Salajegheh, E.; Salajegheh, J. An efficient hybrid of elephant herding optimization and cultural algorithm for optimal design of trusses. *Eng. Comput.* **2018**, *35*, 781–801. [CrossRef]
170. Milani, A.; Santucci, V. Community of scientist optimization: An autonomy oriented approach to distributed optimization. *AI Commun.* **2012**, *25*, 16. [CrossRef]

mathematics

MDPI

Article

Success History-Based Adaptive Differential Evolution Using Turning-Based Mutation

Xingping Sun, Linsheng Jiang, Yong Shen *, Hongwei Kang * and Qingyi Chen

School of Software, Yunnan University, Kunming 650000, China; sunxp@ynu.edu.cn (X.S.); jls@mail.ynu.edu.cn (L.J.); devas9@ynu.edu.cn (Q.C.)
* Correspondence: sheny@ynu.edu.cn (Y.S.); hwkang@ynu.edu.cn (H.K.)

Received: 24 August 2020; Accepted: 7 September 2020; Published: 11 September 2020

Abstract: Single objective optimization algorithms are the foundation of establishing more complex methods, like constrained optimization, niching and multi-objective algorithms. Therefore, improvements to single objective optimization algorithms are important because they can impact other domains as well. This paper proposes a method using turning-based mutation that is aimed to solve the problem of premature convergence of algorithms based on SHADE (Success-History based Adaptive Differential Evolution) in high dimensional search space. The proposed method is tested on the Single Objective Bound Constrained Numerical Optimization (CEC2020) benchmark sets in 5, 10, 15, and 20 dimensions for all SHADE, L-SHADE, and jSO algorithms. The effectiveness of the method is verified by population diversity measure and population clustering analysis. In addition, the new versions (Tb-SHADE, TbL-SHADE and Tb-jSO) using the proposed turning-based mutation get apparently better optimization results than the original algorithms (SHADE, L-SHADE, and jSO) as well as the advanced DISH and the jDE100 algorithms in 10, 15, and 20 dimensional functions, but only have advantages compared with the advanced j2020 algorithm in 5 dimensional functions.

Keywords: single objective optimization; differential evolution; success-history; premature convergence; turning-based mutation

1. Introduction

The single objective global optimization problem involves finding a solution vector $\boldsymbol{x} = (x_1, \ldots, x_D)$ that minimizes the objective function $f(\boldsymbol{x})$, where D is the dimension of the problem. The task of black box optimization is to solve the global optimization problem without clear objective function form or structure, that is, f is a "black box". This problem appears in many problems of engineering optimization, where complex simulations are used to calculate the objective function.

The differential evolution (DE) algorithm, proposed by Price and Storm in 1995, laid the foundation for a series of successful algorithms for continuous optimization. DE is a random black box search method, which was originally designed for numerical optimization problems [1], and it's also an evolutionary algorithm that ensures that every next generation has better solutions than the previous generation: a phenomenon known as elitism. The extensive study fields of DE are summarized lately in the references [2].

Studies on DE have yielded a number of improvements [3–17] to the classical DE algorithm, and the status of research on it can be easily obtained by noting the results of the Continuous Optimization Competition and the Evolutionary Computing Conference (CEC).

A popular variant of DE [18] is the algorithm proposed by Fukunaga and Tanabe called Success History-based Adaptive Differential Evolution (SHADE) [19]. In the optimization process, the scale factor F and the crossover rate CR of control parameters are adjusted to adapt to the given problem, and the "current to $p$best/1" mutation strategy and the external archive of poor quality solutions in

JADE [20] are combined in SHADE. The SHADE algorithm ranked third in CEC2013. In the second year, the author proposed an improved scheme, adding a linear reduction to the population size called L-SHADE to improve the convergence rate of SHADE [21]. L-SHADE won the CEC2014 competition. The winners in the subsequent years were SPS-L-SHADE-EIG [22] (CEC2015), LSHADE-EpSin [23] (joint winner of CEC2016), and jSO [24] (CEC2017). These algorithms are all based on L-SHADE, which makes it one of the most effective variants of SHADE [25]. With the exception of the jSO, the other winners benefited from general enhancements in the area [26]. Consequently, this study applies an improved method to the SHADE, L-SHADE and jSO algorithms. LSHADE-ESP [27] came in second in CEC2018 and the jDE100 [28] won CEC2019. And the j2020 [29] algorithm, which was proposed on CEC2020 recently, is also within the reference range. Enhanced versions of these DE algorithms add new mechanisms or parameters for optimization, similar to those in other optimization algorithms [30], as described in substantive surveys of these areas [25,31–37]. Moreover, theoretical analysis supporting DE has also been provided, such as in [38–41].

The DE consists of three main steps: mutation, crossover, and selection. Many proposals [6,10,11,14, 24] have been made to improve the mutation process to improve optimization performance. For instance, four strategies of combining mutation and crossover was used in SHADE4 [6], SHADE44 [10] and L-SHADE44 [11] to create a new trial individual and realize an adaptive mechanism. A novel multi-chaotic framework was used in MC-SHADE [14] to generate random numbers for the parent selection process in mutation process. A new weighted mutation strategy with parameterization enhancement was used in jSO [24] to enhance adaptability. This paper also focuses on improving this process in the DE algorithm, especially SHADE-based algorithms.

The CEC2020 [42] single-objective boundary-constrained numerical optimization benchmark sets are designed to determine the improvement in performance obtained by increasing the number of the calculation of the fitness function of an optimization algorithm. There are thus two motivations for this study. First, we need to solve the problem of premature convergence of algorithms based on SHADE in high dimensional search spaces on CEC2020 benchmark sets, so that they can maintain a high population diversity and a longer exploration phase. Second, the improvement to the algorithm should be simple, should not excessively increase complexity, and should not render the proposed algorithm incomprehensible and less applicable, as discussed in [43]. We proposed a method using turning-based mutation, and apply it to the SHADE, L-SHADE, and jSO algorithms to yield good performance while using relatively simple algorithm structure. Through experimental analysis involving 10, 15, and 20 dimensions, the improved algorithms achieved better performance than the original algorithms as well as the advanced DISH [44] and jDE100 algorithms on CEC2020 benchmark sets, but were slightly worse than the j2020 algorithm. We also use population diversity measure and population clustering analysis to verify the effectiveness of the proposed method.

Section 2 describes the process of evolution from the DE algorithm to the SHADE, L-SHADE, and jSO algorithms, and turning-based mutation is introduced in Section 3. The experimental settings and results are described in Sections 4 and 5, respectively. Section 6 discusses the results, and the conclusion of this paper is given in Section 7.

2. DE SHADE L-SHADE and jSO

2.1. Differential Evolution

The DE consists of three main steps: mutation, crossover, and selection. In mutation, the attribute vector of the selected individual $\boldsymbol{x}$ is combined in a simple vector operation to generate the mutated vector $\boldsymbol{v}$. The scale factor F of the control parameter is used in this operation. In the crossover step, according to the probability given by the crossover rate CR of the control parameter, the trial vector $\boldsymbol{u}$ is created by selecting the attribute from the original vector $\boldsymbol{x}$ or mutated vector $\boldsymbol{v}$. Finally, in the selection step, the trial vector $\boldsymbol{u}$ is evaluated by the objective function and the fitness $f(\boldsymbol{u})$ is compared

with the fitness of the selected vector $f(\boldsymbol{x})$. The vector with the better fitness value survives to the next generation.

This paper focuses on improving the mutation process, so the paragraphs below describe the mutation process of the DE algorithm. The complete steps of DE can be referred to the literature [1]. The mutation strategy of DE/rand/1/bin can be expressed as follows:

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{r1,G} + F_i \times (\boldsymbol{x}_{r2,G} - \boldsymbol{x}_{r3,G}) \tag{1}$$

where $\boldsymbol{v}_{i,G}$ is the mutated vector, and $\boldsymbol{x}_{r1,G}$, $\boldsymbol{x}_{r2,G}$, and $\boldsymbol{x}_{r3,G}$ are three different individuals randomly selected from the population. F_i is the scaling factor, and G is the index of the current generation.

If any dimension of the mutated vector $\boldsymbol{v}_{j,i,G}$ is outside the boundary of the search range $[x_{min}, x_{max}]$, we perform the following correction for boundary-handling to handle infeasible solutions [45]:

$$\boldsymbol{v}_{j,i,G} = \begin{cases} \frac{x_{min}+\boldsymbol{x}_{j,i,G}}{2} & if\ \boldsymbol{v}_{j,i,G} < x_{min} \\ \frac{x_{max}+\boldsymbol{x}_{j,i,G}}{2} & if\ \boldsymbol{v}_{j,i,G} > x_{max} \end{cases} \tag{2}$$

where j is the dimensional index and i is the individual index.

The pseudo-code of the DE/rand/1/bin algorithm is shown in Algorithm 1.

Algorithm 1 DE/rand/1/bin

```
initialize P, NP, F, CR and MaxFES;
while FES < MaxFES do
  for each individual x do
    use mutation Formula (1) to create mutated vector v;
    execute boundary-handling (2) to handle infeasible solutions;
    use binomial crossover to create trial vector u;
    use selection of classical DE to create individual of next generation;
  end for
end while
return the best found solution.
```

It can be seen from the description of DE algorithm that users need to set three control parameters: crossover rate CR, scaling factor F and population size NP. The setting of these parameters is very important to the performance of DE.

Fine-tuning the control parameter is a time-consuming task, because of which most advanced variants of DE use parameter adaptation. This is also why Tanabe and Fukunaga proposed the SHADE [19] algorithm in 2013. Because the algorithms used in this paper are based on SHADE, it is described in more detail below.

2.2. SHADE

In the control parameters of SHADE, crossover rate CR and scaling factor F are discussed. The algorithm is based on JADE [20], proposed by Sanderson and Zhang, and so they share many mechanisms [18]. The major difference between them is in historical memories $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$ with their update mechanisms. The next subsections describe the historical memory update of SHADE and the difference between DE and SHADE algorithm in initialization, mutation, crossover and selection, respectively. The complete steps of SHADE can be referred to the literature [19].

2.2.1. Initialization

In SHADE, the population is initialized in the same manner as in DE, but there are two additional components—historical memory and external archive—that also need to be initialized.

Initialize the control parameters stored in the historical memory, crossover rate CR and scale factor F to 0.5:

$$\boldsymbol{M}_{CR,i} = \boldsymbol{M}_{F,i} = 0.5; \forall i = 1, \cdots, H, \tag{3}$$

where H is the size of the user-defined historical memory, and the index k to update the historical memory is initialized to one.

In addition, the initialization of the external archive of poor quality solutions is empty, i.e., $\boldsymbol{A} = \varnothing$.

2.2.2. Mutation

In contrast to DE/rand/1/bin, the "current to $p$best/1" mutation strategy is used in SHADE:

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{i,G} + F_i \times \left(\boldsymbol{x}_{pbest,G} - \boldsymbol{x}_{i,G}\right) + F_i \times \left(\boldsymbol{x}_{r1,G} - \boldsymbol{x}_{r2,G}\right) \tag{4}$$

$$p_i = rand[p_{min}, 0.2] \tag{5}$$

$$p_{min} = \frac{2}{NP} \tag{6}$$

where, $\boldsymbol{x}_{i,G}$ is the given individual, and $\boldsymbol{x}_{pbest,G}$ is an individual selected from the best $NP \times p_i$ ($p_i \in [0, 1]$) individuals randomly in the current population. Vector $\boldsymbol{x}_{r1,G}$ is an individual selected from the current population randomly, and $\boldsymbol{x}_{r2,G}$ is an individual selected from a combination of the external archive $\boldsymbol{A}$ and the current population randomly. Index $r1 \neq r2 \neq i$. F_i is a scaling factor, *rand* [] is a uniform random distribution and NP is the size of population. The $\boldsymbol{v}_{i,G}$ is the mutated vector and G is the index of the current generation. The greed of the "current-to-$p$best/1" mutation strategy depends on the control parameter p_i, which is calculated as shown in Equations (5) and (6). It balances exploration and exploitation capabilities (a small value of p is more greedy). The scaling factor F_i is generated using the following formula:

$$F_i = randc_i(\boldsymbol{M}_{F,ri}, 0.1) \tag{7}$$

where $randc_i$ () is the Cauchy distribution, and $\boldsymbol{M}_{F,ri}$ is randomly selected from historical memory $\boldsymbol{M}_F$ (index ri is a uniformly distributed random value from [1, H]). If $F_i > 1$, let $F_i = 1$. If $F_i \leq 0$, Equation (7) is repeated to attempt to generate a valid value.

The boundary handling of SHADE is identical to that of DE, as shown in Equation (2).

2.2.3. Crossover

DE has 2 classic crossover strategies, i.e., binomial and exponential. The crossover strategies of SHADE is the same as that of DE/rand/1/bin, i.e., binomial crossover. However, the crossover rate of DE/rand/1/bin is set in advance while the CR_i of SHADE is generated by the following formula:

$$CR_i = randn_i(\boldsymbol{M}_{CR,ri}, 0.1) \tag{8}$$

where $randn_i$ () is Gaussian distribution, and $\boldsymbol{M}_{CR,ri}$ are randomly selected from historical memory $\boldsymbol{M}_{CR}$ (index r_i is a uniformly distributed random value from [1, H]). If $CR_i > 1$, let $CR_i = 1$; if $CR_i < 0$, let $CR_i = 0$.

2.2.4. Selection

The process of selection of SHADE is the same as that of DE. However, the external archive needs to be updated during selection. If a better trail individual is generated, the original individual $\boldsymbol{x}_{i,G}$ is stored in the external archive. If the external archive exceeds capacity, one of them is randomly deleted.

2.2.5. Historical Memory Update

Historical memory update is also an important operation in SHADE. The historical memories $\boldsymbol{M}_{CR}$ and $\boldsymbol{M}_F$ are initialized by Formula (3) but their contents change with the iteration of the algorithm.

These memories store the "successful" crossover rate CR and scaling factor F. "Successful" here means that the trail vector $\boldsymbol{u}$ is selected instead of the original vector $\boldsymbol{x}$ to survive to the next generation. In each generation, the values of these "successful" CR and F are first stored in arrays $\boldsymbol{S}_{CR}$ and $\boldsymbol{S}_F$, respectively. After each generation, a unit of each of the historical memories $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$ is updated. The updated unit is specified by index k, which is initialized to one and increases by one after each generation. If k exceeds the memory capacity H, it is reset to one. The following formula is used to update the k-th unit of historical memory:

$$\boldsymbol{M}_{CR,k,G+1} = \begin{cases} mean_{WA}(\boldsymbol{S}_{CR}) \; if \; \boldsymbol{S}_{CR} \neq \varnothing \\ \boldsymbol{M}_{CR,k,G} \; otherwise \end{cases} \tag{9}$$

$$\boldsymbol{M}_{F,k,G+1} = \begin{cases} mean_{WL}(\boldsymbol{S}_F) \; if \; \boldsymbol{S}_F \neq \varnothing \\ \boldsymbol{M}_{F,k,G} \; otherwise \end{cases} \tag{10}$$

If all individuals in the G-th generation fail to generate a better trail vector, i.e., $\boldsymbol{S}_F = \boldsymbol{S}_{CR} = \emptyset$, the historical memory will not be updated. The weighted Lehmer mean W_L and weighted mean W_A are calculated using the following formulas, respectively:

$$mean_{WA}(\boldsymbol{S}_{CR}) = \sum_{k=1}^{|S_{CR}|} \boldsymbol{w}_k \times \boldsymbol{S}_{CR,k}, \tag{11}$$

$$mean_{WL}(\boldsymbol{S}_F) = \frac{\sum_{k=1}^{|S_F|} \boldsymbol{w}_k \times \boldsymbol{S}_{F,k}^2}{\sum_{k=1}^{|S_F|} \boldsymbol{w}_k \times \boldsymbol{S}_{F,k}} \tag{12}$$

To improve the adaptability of the parameters, the weight vector $\boldsymbol{w}$ is calculated based on the absolute value of the difference that is obtained by subtracting the objective function value of the given vector from that of the trail vector in current generation G, as follows:

$$\boldsymbol{w}_k = \frac{\Delta f_k}{\sum_{k=1}^{|S_{CR}|} \Delta f_k} \tag{13}$$

where $\Delta f_k = \left| f\left(\boldsymbol{u}_{k,G}\right) - f\left(x_{k,G}\right) \right|$ in (13).

The pseudo-code of the SHADE algorithm is shown in Algorithm 2.

Algorithm 2 SHADE

```
initialize P, NP, F, CR, A, H and MaxFES;
initialize M_F, M_CR by (3);
while FES < MaxFES do
    for each individual x do
        use mutation Formulas (7) and (8) to select F and CR;
        use Formula (4) to create mutated vector v;
        execute boundary-handling (2) to handle infeasible solutions;
        use binomial crossover to create trial vector u;
        use selection of classical DE to create individual of next generation;
        update external archive A;
    end for
    use Formulas (9) and (10) to update historical memory M_F and M_CR;
end while
return the best found solution.
```

2.3. Linear Decrease in Population Size: L-SHADE

In [21], a linear reduction of population size was introduced to SHADE to improve its performance. The basic thought is to gradually reduce the population size during evolution to improve exploitation

capabilities. In L-SHADE, the population size is calculated after each generation using Formula (14). If the new population size NP_{new} is smaller than the previous population size NP, the all individuals are sorted on the basis of the value of the objective function, and the worst NP-NP_{new} individuals are cut. Also, the size of external archives/$\boldsymbol{A}$/decreases synchronously with population size:

$$NP_{new} = round\left(NP_{init} - \frac{FES}{MAXFES} \times \left(NP_{init} - NP_f\right)\right) \tag{14}$$

where NP_f and NP_{init} are the final and initial population size, respectively. *MaxFES* and *FES* are the maximum and current number of the calculation of the fitness function, respectively. And *round* () is a rounding function.

2.4. Weighted Mutation Strategy with Parameterization Enhancement: jSO

The jSO [24] algorithm won the CEC2017 single-objective real parameter optimization competition [46]. It is a type of iL-SHADE algorithm that uses a weighted mutation strategy [47]. The iL-SHADE algorithm extends L-SHADE by initializing all parameters in the historical memories $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$ to 0.8, statically initializing the last unit of historical memories $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$ to 0.9, updating $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$ with the weighted Lehmer average value, limiting the crossover rate CR and scaling factor F in the early stage, and p is calculated for the "current-to-$p$best/1" mutation strategy as:

$$p = p_{min} + \frac{FES}{MAXFES}(p_{max} - p_{min}) \tag{15}$$

where p_{min} and p_{max} are the minimum and maximum value of p, respectively. *FES* and *MaxFES* are the current and maximum number of the calculation of the fitness function, respectively.

The jSO algorithm sets $p_{max} = 0.25$ and $p_{min} = p_{max}/2$, initial population size to $NP_{init} = 25\sqrt{D}\log D$, and the size of the historical memory to $H = 5$. All parameters in $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$ are initialized to 0.3 and 0.8, respectively, and the weighted mutation strategy current-to-$p$best-w/1 is used:

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{i,G} + F_w \times \left(\boldsymbol{x}_{pbest,G} - \boldsymbol{x}_{i,G}\right) + F_i \times (\boldsymbol{x}_{r1,G} - \boldsymbol{x}_{r2,G}), \tag{16}$$

where F_w is calculated as:

$$F_w = \begin{cases} 0.7F_i, & FES < 0.2MAXFES, \\ 0.8F_i, & FES < 0.4MAXFES, \\ 1.2F_i, & otherwise. \end{cases} \tag{17}$$

3. Turning-Based Mutation

The opposition-based DE (ODE) algorithm was proposed by Shahryar et al. [48]. The opposition-based learning (OBL) was used for generation jumping and population initialization, and the opposite numbers was used to improve the convergence rate of DE. Shahryar et al. let all vectors of the initial population take the opposite number in the initialization and allowed the trail vectors to take the opposite number in the selection operation. They then compared their fitness values and selected the vector with the better fitness to accelerate the convergence of the DE algorithm. We refer to the idea of "opposition" in the above algorithm, but the purpose of this paper is to change the direction of mutation under certain conditions to maintain population diversity and enable a longer exploration phase.

Suppose that the search space is two-dimensional (2D). There is a ring-shaped region, the center of which is the global suboptimal individual $x_{pbest,G}$. The outer radius of the ring is OR and the inner radius is IR. If the Euclidean distance *Distance* between the given individual and the global suboptimal individual is smaller than the outer radius OR and larger than the inner radius IR, the differential vector de_i from the mutation Formulas (1) and (4) takes the opposite number, and some dimensions are

randomly selected to assign random values within the search range. Experiments have verified that better outer radius *OR* and inner radius *IR* can be calculated as:

$$OR_{init} = \sum_{j=1}^{D} \sqrt[2]{\left(\frac{x_{max} - x_{min}}{2}\right)^2} \tag{18}$$

$$IR = \sum_{j=1}^{D} \sqrt[2]{\left(\frac{x_{max} - x_{min}}{40}\right)^2} \tag{19}$$

$$OR = OR_{init} + \frac{IR - OR_{init}}{MaxFES} \times FES \tag{20}$$

where OR_{init} is the initial value of the outer radius and *IR* is the inner radius, which is also the minimum value of the outer radius. The outer radius *OR* decreases with an increase in the number of fitness evaluations. *MaxFES* and *FES* are the maximum and current number of the calculation of the fitness function, respectively, and x_{max} and x_{min} are the upper and lower bounds of the search range, respectively.

The Euclidean distance *Distance* between the given individual and the global suboptimal individual is calculated as:

$$Distance = \sqrt[2]{\sum_{j=1}^{D} \left(\boldsymbol{x}_{j,pbest,G} - \boldsymbol{x}_{j,i,G}\right)^2} \tag{21}$$

The differential vector de_i from the mutation Equations (1) and (4) is calculated as:

$$de_i = (F_i \; or \; F_w) \times \left(\boldsymbol{x}_{pbest,G} - \boldsymbol{x}_{i,G}\right) + F_i \times \left(\boldsymbol{x}_{r1,G} - \boldsymbol{x}_{r2,G}\right) \tag{22}$$

The pseudo-code of the operation on the differential vector de_i in turning-based mutation is shown as Operation 1:

Operation 1 operation on de_i

```
if Distance > IR and Distance < OR then
    de_i = −de_i;
    M = randi(D), R = randperm(D);
    for d = 1 to M do
        de_i(R(d)) = rand (x_max−x_min) + x_min;
    end for
end if
```

where $\boldsymbol{R}$ is the randomly disordered dimension index array, *M* is the number of randomly selected dimensions, and x_{max} and x_{min} are the upper and lower bounds of the search range, respectively.

Finally, the mutation operation is performed as shown in Equation (23):

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{i,G} + de_i \tag{23}$$

If the Euclidean distance *Distance* between the given individual and the global suboptimal individual is smaller than the outer radius *OR* and larger than the inner radius *IR*, the improved method changes the direction of mutation of the given individual to maintain the population diversity and a longer exploration phase, thus enhancing the global search ability and the ability to escape the local optimum. Then, with an increase in number of fitness evaluations, the performance of the algorithm can be improved. If the Euclidean distance *Distance* between the given individual and the global suboptimal individual is smaller than or equal to the inner radius *IR*, the former is

allowed to mutate in the original direction. This enables the given individual to quickly converge to the global optimal or suboptimal position to avoid the problem of non-convergence caused by turning-based mutation.

Since Equation (21) and Operation 1 need to be executed in the mutation process of each individual, the overall time complexity [42] of the improved algorithms is slightly higher than that of the original algorithms, as shown in Tables 1–3.

Table 1. Time complexity specified by CEC2020 technical document-SHADE vs. Tb-SHADE.

D	T0	T1	SHADE		Tb-SHADE	
			T2	(T2 − T1)/T0	T2	(T2 − T1)/T0
5	6.03E+01	2.52E+02	4.81E+03	7.56E+01	5.58E+03	8.84E+01
10	6.03E+01	3.05E+02	5.55E+03	8.70E+01	6.35E+03	1.00E+02
15	6.03E+01	3.39E+02	5.68E+03	8.86E+01	6.76E+03	1.06E+02
20	6.03E+01	4.09E+02	6.03E+03	9.32E+01	7.14E+03	1.12E+02

Table 2. Time complexity specified by CEC2020 technical document—L-SHADE vs. TbL-SHADE.

D	T0	T1	L-SHADE		TbL-SHADE	
			T2	(T2 − T1)/T0	T2	(T2 − T1)/T0
5	6.03E+01	2.52E+02	4.44E+03	6.95E+01	4.97E+03	7.82E+01
10	6.03E+01	3.05E+02	4.77E+03	7.40E+01	6.71E+03	1.06E+02
15	6.03E+01	3.39E+02	4.98E+03	7.70E+01	6.97E+03	1.10E+02
20	6.03E+01	4.09E+02	5.24E+03	8.01E+01	7.30E+03	1.14E+02

Table 3. Time complexity specified by CEC2020 technical document-jSO vs. Tb-jSO.

D	T0	T1	jSO		Tb-jSO	
			T2	(T2 − T1)/T0	T2	(T2 − T1)/T0
5	6.03E+01	2.52E+02	4.30E+03	6.71E+01	5.24E+03	8.27E+01
10	6.03E+01	3.05E+02	5.28E+03	8.25E+01	6.51E+03	1.03E+02
15	6.03E+01	3.39E+02	6.50E+03	1.02E+02	7.12E+03	1.12E+02
20	6.03E+01	4.09E+02	7.27E+03	1.14E+02	7.81E+03	1.23E+02

The pseudo-code of the Tb-SHADE algorithm (SHADE algorithm using turning-based mutation) is shown as Algorithm 3, that of the TbL-SHADE algorithm (L-SHADE algorithm using turning-based mutation) is shown as Algorithm 4, and that of the Tb-jSO algorithm (jSO algorithm using turning-based mutation) is shown as Algorithm 5. The improved parts of these algorithm are underlined.

Algorithm 3 Tb-SHADE

initialize $\boldsymbol{P}$, NP, F, CR, $\boldsymbol{A}$, H and $MaxFES$;
initialize $\boldsymbol{M}_F$, $\boldsymbol{M}_{CR}$ by (3);
initialize OR_{init} by (18), initialize IR by (19);
while $FES < MaxFES$ **do**
  Calculate OR by (20);
  for each individual x **do**
    use mutation Formulas (7) and (8) to select F and CR;
    Set $Distance$ by (21);
    Execute Operation 1;
    use Formula (23) to create mutated vector v;
    execute boundary-handling (2) to handle infeasible solutions;
    use binomial crossover to create trial vector $\boldsymbol{u}$;
    use selection of classical DE to create individual of next generation;
    update external archive $\boldsymbol{A}$;
  end for
  use Formulas (9) and (10) to update historical memory $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$;
end while
return the best found solution.

Algorithm 4 TbL-SHADE

initialize $\boldsymbol{P}$, NP_{init}, NP_f, F, CR, $\boldsymbol{A}$, H and $MaxFES$;
initialize $\boldsymbol{M}_F$, $\boldsymbol{M}_{CR}$ by (3);
initialize OR_{init} by (18), initialize IR by (19);
while $FES < MaxFES$ **do**
  Calculate OR by (20);
  for each individual x **do**
    use mutation Formulas (7) and (8) to select F and CR;
    Set $Distance$ by (21);
    Execute Operation 1;
    use Formula (23) to create mutated vector v;
    execute boundary-handling (2) to handle infeasible solutions;
    use binomial crossover to create trial vector $\boldsymbol{u}$;
    use selection of classical DE to create individual of next generation;
    update external archive $\boldsymbol{A}$;
  end for
  use Formulas (9) and (10) to update historical memory $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$;
use (14) to calculate NP_{new};
$NP = NP_{\text{new}}$, $|\boldsymbol{A}| = NP_{\text{new}}$;
end while
return the best found solution.

Algorithm 5 Tb-jSO

initialize $\boldsymbol{P}$, NP_{init}, NP_f, F, CR, $\boldsymbol{A}$, H and $MaxFES$;
initialize all values in $\boldsymbol{M}_F$ to 0.3 and $\boldsymbol{M}_{CR}$ to 0.8, but $\boldsymbol{M}_{F,H} = 0.9$ and $\boldsymbol{M}_{CR,H} = 0.9$;
initialize OR_{init} by (18), initialize IR by (19);
while $FES < MaxFES$ **do**
  Calculate OR by (20);
  for each individual $\boldsymbol{x}$ **do**
    use mutation Formulas (7) and (8) to select F and CR;
    use (17) to calculate F_w;
    if $FES < 0.6MaxFES$ **and** $F_{i,G} > 0.7$ **then**
      $F_{i,G} = 0.7$;
    end if
    if $FES < 0.25MaxFES$ **then**
      $CR_{i,G} = \max(CR_{i,G}, 0.7)$;
    else if $FES < 0.5MaxFES$ **then**
      $CR_{i,G} = \max(CR_{i,G}, 0.6)$;
    end if
    Set $Distance$ by (21);
    Execute Operation 1;
    use Formula (23) to create mutated vector $\boldsymbol{v}$;
    execute boundary-handling (2) to handle infeasible solutions;
    use binomial crossover to create trial vector $\boldsymbol{u}$;
    use selection of classical DE to create individual of next generation;
    update external archive $\boldsymbol{A}$;
  end for
  use Formulas (9) and (10) to update historical memory $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$;
  use (14) to calculate NP_{new};
  $NP = NP_{\text{new}}$, $|\boldsymbol{A}| = NP_{\text{new}}$;
end while
return the best found solution.

4. Experimental Settings

To verify the improved method by experiments, the original algorithm, the improved algorithm and the advanced DISH and the jDE100 algorithms were tested on the Single Objective Bound Constrained Numerical Optimization (CEC2020) benchmark sets in 5, 10, 15 and 20 dimensions. The termination criteria, i.e., the maximum number of the calculation of the fitness function (*MaxFES*) and the minimum error value (*Min error value*), were set as in Table 4. The search range is $[x_{min}, x_{max}] = [-100, 100]$, and 30 independent repeated experiments were conducted. The parameter setting of most algorithm [19,21,24] is shown in Tables 5 and 6. In addition, the parameter setting of j2020 algorithm can be found in [29].

Table 4. Termination criteria.

D	*MaxFES*	*Min Error Value*
5	50,000	10^{-8}
10	1,000,000	10^{-8}
15	3,000,000	10^{-8}
20	10,000,000	10^{-8}

Table 5. Parameter setting of some algorithms.

Algorithm	NP	H	$\|A\|$	NP_{init}	NP_f	$MaxG$	M_{CRinit}	M_{Finit}
SHADE	100	NP	NP	–	–	$MaxFES/NP$	0.5	0.5
Tb-SHADE	100	NP	NP	–	–	$MaxFES/NP$	0.5	0.5
L-SHADE	Calculated by (18)	100	NP	100	4	Not fixed	0.5	0.5
TbL-SHADE	Calculated by (18)	100	NP	100	4	Not fixed	0.5	0.5
jSO	Calculated by (18)	5	NP	25logD	4	Not fixed	0.8 $M_{CR,H} = 0.9$	0.3 $M_{F,H} = 0.9$
Tb-jSO	Calculated by (18)	5	NP	25logD	4	Not fixed	0.8 $M_{CR,H} = 0.9$	0.3 $M_{F,H} = 0.9$
DISH	Calculated by (18)	5	NP	25logD	4	Not fixed	0.8 $M_{CR,H} = 0.9$	0.3 $M_{F,H} = 0.9$

Table 6. Parameter setting of jDE100.

Parameter	Value	Description
F_l	$\frac{5.0}{\sqrt{bNP}}$	lower limit of scale factor for the big population
F_l	$\frac{1.0}{\sqrt{bNP}}$	lower limit of scale factor for the small population
F_u	1.1	upper limit of scale factor
CR_l	0.0	lower limit of crossover parameter
CR_u	1.1	upper limit of crossover parameter
F_{init}	0.5	initial value of scale factor
CR_{init}	0.5	initial value of crossover parameter
τ_1	0.1	probability to self-adapt scale factor
τ_2	0.1	probability to self-adapt crossover parameter
bNP	1000	size of big population
sNP	25	size of small population
$ageLmt$	1×10^9	number of FEs when population restart needs to occurs
eps	1×10^{-16}	small value used to check if two value are similar
$myEqs$	25	reinitialization if myEqs% of individuals in the corresponding population have the similar function values
$MaxG$	Not fixed	the maximum number of generations

The hypothesis that the turning-based mutation can maintain a longer exploration phase can be verified by analyzing the clustering and density of the population during the optimization process. These two analyses are described in more detail below.

4.1. Cluster Analysis

The clustering algorithm selected in this experiment is density based noisy application spatial clustering (DBSCAN) [49], which is based on the clustering density rather than its center, so it can find clusters of arbitrary shape. DBSCAN algorithm needs to set two control parameters and a distance measurement. The settings are as follows:

(1) distance between core points, that is, Eps = 1% of the decision space; for the CEC2020 benchmark sets, Eps = 2;

(2) minimum number of points forming a cluster, that is, $MinPts$ = 4 (minimum number of mutation individuals); and

(3) distance measure uses Chebyshev distance [50]—if the distance between the corresponding attributes of two individuals is greater than 1% of the decision space, they are not considered as direct density reachable.

4.2. Population Diversity

The population diversity (PD) measure is taken from [51], which is based on the square root of the deviation sum, Equation (25), of individual components and their corresponding mean values, Equation (24):

$$\overline{x_j} = \frac{1}{NP}\sum_{i=1}^{NP} \boldsymbol{x}_{j,i} \tag{24}$$

$$PD = \sqrt{\frac{1}{NP}\sum_{i=1}^{NP}\sum_{j=1}^{D}\left(\boldsymbol{x}_{j,i} - \overline{x_j}\right)^2}. \tag{25}$$

where i is the iterator of members of the population and j is that of the component (dimension).

5. Results

Tables 7–18 compare the error values (when the error value was smaller than 10^{-8}, the corresponding value was considered optimal) obtained by the original algorithms (SHADE, L-SHADE, and jSO) and their improved versions using turning-based mutation (Tb-SHADE, TbL-SHADE and Tb-jSO, respectively). The results of a comparison are showed in the last column of each table. If the performance of the original version was significantly better, uses the "−" sign; if the performance of the improved version was significantly better, uses the sign "+"; if their performances were similar, "=" is used. The better performance values are displayed in bold, and the last row of these tables shows the results of an overall comparison. Tables 19–22 provide the error values obtained by the advanced algorithms DISH, jDE100 and j2020 on CEC2020. All tables provide the best, mean and std (standard deviation) values of 30 independent repetitions of the experiments.

Table 7. SHADE vs. Tb-SHADE on CEC2020 in 5*D*.

f	SHADE			Tb-SHADE			Result
	Best	**Mean**	**Std**	**Best**	**Mean**	**Std**	
1	1.70E−09	6.44E−09	2.40E−09	8.41E−01	5.37E+00	3.83E+00	−
2	3.78E−01	1.73E+00	1.80E+00	3.43E+00	1.30E+01	6.19E+00	−
3	1.32E+00	5.07E+00	1.07E+00	1.42E+00	6.73E+00	2.16E+00	=
4	1.23E−02	9.32E−02	4.43E−02	4.30E−02	1.71E−01	8.96E−02	=
5	1.65E−09	6.89E−09	2.39E−09	3.71E−02	1.05E+00	6.62E−01	=
6	5.41E−10	6.48E−09	2.54E−09	2.49E−02	5.89E−02	1.81E−02	=
7	2.83E−10	6.62E−09	3.01E−09	1.32E−05	2.12E−04	6.34E−04	=
8	4.29E−09	3.34E+00	1.83E+01	4.09E−01	2.23E+00	1.61E+00	=
9	1.00E+02	1.07E+02	3.65E+01	2.07E+01	7.80E+01	2.78E+01	+
10	3.00E+02	3.46E+02	8.65E+00	5.62E+01	2.76E+02	7.49E+01	+
				2 + 2 −			

Table 8. SHADE vs. Tb-SHADE on CEC2020 in 10*D*.

f	SHADE			Tb-SHADE			Result
	Best	**Mean**	**Std**	**Best**	**Mean**	**Std**	
1	2.12E−09	7.52E−09	2.16E−09	1.50E−09	7.38E−09	2.13E−09	=
2	1.91E−01	5.14E+00	4.17E+00	3.67E+00	2.26E+01	1.22E+01	−
3	1.05E+01	1.18E+01	7.29E−01	3.71E+00	1.03E+01	3.49E+00	+
4	9.90E−02	1.63E−01	2.75E−02	3.67E−05	2.54E−02	1.47E−02	=
5	8.82E−09	3.54E−01	1.91E−01	9.23E−07	5.48E+00	5.13E+00	−
6	7.17E−02	1.95E−01	1.01E−01	1.68E−01	5.28E−01	2.05E−01	=
7	4.09E−06	1.39E−01	1.76E−01	3.97E−02	1.83E−01	1.19E−01	=
8	1.00E+02	1.00E+02	6.24E−07	2.15E−04	1.86E+01	1.76E+01	+
9	1.00E+02	2.92E+02	8.72E+01	1.00E+02	1.00E+02	4.45E−02	+
10	3.98E+02	4.16E+02	2.29E+01	1.00E+02	1.21E+02	6.61E+01	+
				4 + 2 −			

Table 9. SHADE vs. Tb-SHADE on CEC2020 in 15*D*.

f	SHADE			Tb-SHADE			Result
	Best	Mean	Std	Best	Mean	Std	
1	6.48E−09	8.35E−09	9.82E−10	2.06E−09	7.43E−09	2.01E−09	=
2	1.68E−01	7.53E+00	2.16E+01	2.46E+00	3.08E+01	2.79E+01	−
3	1.56E+01	1.57E+01	2.05E−01	3.64E+00	8.46E+00	2.91E+00	+
4	1.78E−01	2.74E−01	3.69E−02	4.06E−02	7.41E−02	3.22E−02	=
5	1.31E+00	5.07E+01	5.74E+01	2.41E+01	4.89E+01	1.76E+01	−
6	7.43E−02	3.78E−01	2.22E−01	3.26E−01	2.61E+00	2.98E+00	=
7	4.18E−01	2.06E+01	4.50E+01	3.02E−01	3.07E+00	2.01E+00	+
8	1.00E+02	1.00E+02	0.00E+00	8.74E−09	5.20E+01	4.49E+01	+
9	3.38E+02	3.87E+02	9.71E+00	1.00E+02	1.46E+02	1.01E+02	+
10	4.00E+02	4.00E+02	0.00E+00	1.00E+02	2.07E+02	7.85E+01	+
				5 + 2 −			

Table 10. SHADE vs. Tb-SHADE on CEC2020 in 20*D*.

f	SHADE			Tb-SHADE			Result
	Best	Mean	Std	Best	Mean	Std	
1	5.87E−09	8.38E−09	9.25E−10	2.21E−09	7.94E−09	1.76E−09	=
2	8.52E−09	2.49E−01	5.14E−01	9.38E−02	1.97E+00	1.38E+00	=
3	2.04E+01	2.06E+01	3.27E−01	6.66E−09	1.46E+01	8.57E+00	+
4	2.27E−01	3.78E−01	4.93E−02	3.50E−01	4.09E−01	3.48E−02	=
5	2.06E+01	2.04E+02	8.51E+01	7.38E+00	1.56E+02	1.11E+02	+
6	9.26E−02	2.04E−01	6.62E−02	2.62E−01	3.84E−01	6.17E−02	=
7	3.55E−01	4.53E+01	5.54E+01	3.06E+00	2.49E+01	2.23E+01	+
8	1.00E+02	1.00E+02	2.23E−13	1.00E+02	1.00E+02	1.89E−13	=
9	4.01E+02	4.05E+02	2.00E+00	1.00E+02	3.99E+02	6.89E+01	+
10	4.14E+02	4.14E+02	9.83E−03	4.10E+02	4.13E+02	9.93E−01	=
				4 + 0 −			

Table 11. L-SHADE vs. TbL-SHADE on CEC2020 in 5*D*.

f	L-SHADE			TbL-SHADE			Result
	Best	Mean	Std	Best	Mean	Std	
1	3.32E−09	7.16E−09	1.95E−09	1.86E−09	5.06E−05	2.77E−04	=
2	6.54E−10	9.34E−02	1.14E−01	1.30E−05	4.24E−01	1.22E+00	=
3	5.15E+00	5.17E+00	6.51E−02	6.14E−01	2.96E+00	1.73E+00	+
4	9.92E−03	6.52E−02	3.08E−02	7.35E−07	1.58E−02	1.81E−02	=
5	2.24E−09	6.88E−09	2.02E−09	2.19E−09	3.60E−05	1.97E−04	=
6	8.44E−10	5.15E−09	2.84E−09	6.25E−11	6.00E−09	5.02E−09	=
7	1.51E−09	6.13E−09	2.69E−09	7.16E−10	4.76E−09	2.96E−09	=
8	5.19E−09	7.95E−09	1.44E−09	6.25E−09	2.77E−04	1.04E−03	=
9	7.81E−09	9.67E+01	1.83E+01	4.50E−09	6.61E+01	4.36E+01	+
10	3.00E+02	3.44E+02	1.20E+01	8.19E−09	2.71E+02	8.34E+01	+
				3 + 0 −			

Table 12. L-SHADE vs. TbL-SHADE on CEC2020 in 10*D*.

f	L-SHADE			TbL-SHADE			Result
	Best	Mean	Std	Best	Mean	Std	
1	5.88E−09	8.29E−09	1.22E−09	1.70E−09	6.65E−09	2.33E−09	=
2	3.44E−04	1.13E+00	1.53E+00	6.39E−02	7.09E+00	8.75E+00	−
3	1.04E+01	1.07E+01	2.92E−01	4.00E+00	9.40E+00	3.00E+00	+
4	9.87E−02	1.52E−01	2.23E−02	9.53E−09	1.90E−02	1.10E−02	=
5	3.83E−09	4.32E+00	2.20E+01	5.39E−02	1.60E+00	1.25E+00	=
6	2.33E−02	9.59E−02	5.58E−02	7.39E−02	3.75E−01	1.38E−01	=
7	1.06E−07	1.39E−01	2.02E−01	1.26E−06	2.14E−03	2.71E−03	=
8	7.59E−09	9.67E+01	1.83E+01	9.62E−09	3.20E+01	2.15E+01	+
9	1.00E+02	2.91E+02	8.70E+01	5.77E−09	1.02E+02	5.05E+01	+
10	3.98E+02	4.16E+02	2.29E+01	1.00E+02	1.40E+02	1.03E+02	+
				4 + 1 −			

Table 13. L-SHADE vs. TbL-SHADE on CEC2020 in 15*D*.

f	L-SHADE			TbL-SHADE			Result
	Best	Mean	Std	Best	Mean	Std	
1	3.53E−09	7.72E−09	1.70E−09	1.35E−09	7.01E−09	2.54E−09	=
2	3.64E−12	3.73E−01	7.99E−01	8.33E−02	1.08E+00	1.74E+00	=
3	1.56E+01	1.56E+01	1.41E−01	4.45E−09	2.55E+00	1.55E+00	+
4	2.07E−01	2.64E−01	3.91E−02	9.87E−03	4.13E−02	1.23E−02	=
5	2.46E+00	5.21E+01	6.26E+01	7.51E+00	3.40E+01	1.58E+01	+
6	2.43E−03	4.23E−01	1.52E+00	1.13E−01	1.82E+00	3.20E+00	=
7	6.63E−02	1.65E+01	4.13E+01	4.35E−01	9.54E−01	3.53E−01	+
8	1.00E+02	1.00E+02	0.00E+00	7.71E−09	6.02E+01	4.20E+01	+
9	3.00E+02	3.80E+02	2.36E+01	1.00E+02	1.80E+02	1.27E+02	+
10	4.00E+02	4.00E+02	0.00E+00	1.00E+02	2.20E+02	1.27E+02	+
				6 + 0 −			

Table 14. L-SHADE vs. TbL-SHADE on CEC2020 in 20*D*.

f	L-SHADE			TbL-SHADE			Result
	Best	Mean	Std	Best	Mean	Std	
1	4.35E−09	8.54E−09	1.33E−09	1.50E−09	7.72E−09	2.26E−09	=
2	9.46E−11	3.44E−02	3.88E−02	3.12E−02	5.21E−01	7.45E−01	=
3	2.04E+01	2.06E+01	3.93E−01	1.74E−09	2.28E+00	1.58E+00	+
4	2.57E−01	3.66E−01	3.95E−02	2.97E−02	7.35E−02	2.85E−02	=
5	3.02E+01	2.46E+02	1.02E+02	1.13E+01	2.42E+02	1.39E+02	+
6	8.46E−02	1.85E−01	7.59E−02	1.97E−01	3.13E−01	5.12E−02	=
7	2.23E−01	4.34E+01	5.64E+01	1.58E+00	4.15E+01	3.84E+01	=
8	1.00E+02	1.00E+02	1.89E−13	5.14E+01	9.40E+01	1.45E+01	+
9	4.00E+02	4.04E+02	2.37E+00	1.00E+02	4.12E+02	9.59E+01	+
10	4.14E+02	4.14E+02	1.08E−02	3.99E+02	4.03E+02	4.09E+00	+
				5 + 0 −			

Table 15. jSO vs. Tb-jSO on CEC2020 in 5*D*.

f	jSO			Tb-jSO			Result
	Best	Mean	Std	Best	Mean	Std	
1	1.34E−09	6.70E−09	2.16E−09	2.24E−09	7.69E−09	2.26E−09	=
2	9.71E−09	4.11E−01	1.24E+00	8.54E−09	2.01E+00	3.43E+00	=
3	6.13E−01	4.92E+00	1.22E+00	2.91E−08	2.34E+00	1.75E+00	=
4	8.28E−09	6.28E−02	3.35E−02	1.92E−09	3.03E−02	3.47E−02	=
5	3.14E−09	2.08E−02	1.14E−01	3.85E−09	7.48E−02	2.35E−01	=
6	1.00E−09	5.84E−09	2.36E−09	4.08E−10	6.74E−09	2.84E−09	=
7	1.57E−10	5.51E−09	2.94E−09	3.82E−11	4.59E−09	3.30E−09	=
8	4.22E−09	7.95E−09	1.63E−09	5.87E−09	8.58E−09	1.08E−09	=
9	5.55E−09	9.67E+01	1.83E+01	6.29E−09	9.33E+01	2.54E+01	=
10	3.00E+02	3.46E+02	8.65E+00	3.00E+02	3.08E+02	1.80E+01	+
				1 + 0 −			

Table 16. jSO vs. Tb-jSO on CEC2020 in 10*D*.

f	jSO			Tb-jSO			Result
	Best	Mean	Std	Best	Mean	Std	
1	4.05E−09	7.91E−09	1.39E−09	4.56E−09	8.54E−09	1.28E−09	=
2	3.12E−01	6.99E+00	4.53E+00	3.54E+00	2.03E+01	2.14E+01	−
3	1.04E+01	1.19E+01	6.21E−01	2.62E+00	8.28E+00	2.85E+00	+
4	9.86E−02	1.58E−01	3.20E−02	1.97E−02	5.51E−02	3.94E−02	=
5	6.31E−09	2.61E−01	2.94E−01	9.79E−09	1.50E+00	9.92E−01	=
6	1.95E−02	1.05E−01	8.44E−02	2.93E−02	3.63E−01	1.75E−01	=
7	6.94E−07	7.13E−02	1.60E−01	1.23E−06	4.87E−02	1.01E−01	=
8	1.00E+02	1.00E+02	0.00E+00	6.98E−09	1.45E+00	4.49E+00	+
9	1.00E+02	3.02E+02	6.89E+01	1.00E+02	1.19E+02	6.22E+01	+
10	3.98E+02	4.04E+02	1.57E+01	1.00E+02	3.88E+02	5.44E+01	+
				4 + 1 −			

Table 17. jSO vs. Tb-jSO on CEC2020 in 15*D*.

f	jSO			Tb-jSO			Result
	Best	Mean	Std	Best	Mean	Std	
1	4.88E−09	8.28E−09	1.34E−09	4.93E−09	8.62E−09	1.36E−09	=
2	4.16E−02	2.61E+01	4.47E+01	1.67E−01	2.04E+01	2.49E+01	+
3	1.56E+01	1.66E+01	5.14E−01	1.99E+00	4.75E+00	1.59E+00	+
4	1.78E−01	2.62E−01	3.76E−02	2.96E−02	9.31E−02	4.54E−02	=
5	1.15E+00	3.07E+00	2.07E+00	1.56E−01	1.13E+01	1.04E+01	−
6	3.33E−02	3.20E−01	3.15E−01	8.05E−02	2.70E−01	1.26E−01	=
7	1.24E−01	7.15E−01	2.13E−01	1.06E−01	4.85E−01	2.48E−01	=
8	1.00E+02	1.00E+02	0.00E+00	7.83E−09	5.54E+01	4.08E+01	+
9	3.86E+02	3.89E+02	8.36E−01	1.00E+02	2.55E+02	1.48E+02	+
10	4.00E+02	4.00E+02	0.00E+00	4.00E+02	4.00E+02	0.00E+00	=
				4 + 1 −			

Table 18. jSO vs. Tb-jSO on CEC2020 in 20*D*.

f	jSO			Tb-jSO			Result
	Best	Mean	Std	Best	Mean	Std	
1	5.80E−09	8.53E−09	1.32E−09	6.70E−09	9.21E−09	8.21E−10	=
2	6.25E−02	1.99E+00	1.60E+00	1.74E+00	6.29E+00	3.45E+00	=
3	2.04E+01	2.13E+01	5.24E−01	2.56E+00	5.17E+00	1.63E+00	+
4	1.97E−01	3.53E−01	4.48E−02	5.92E−02	1.21E−01	5.05E−02	=
5	1.20E+00	6.93E+00	5.07E+00	4.16E−01	3.27E+01	4.81E+01	−
6	5.63E−02	9.75E−01	4.25E−01	5.93E−02	3.11E−01	1.49E−01	=
7	5.31E−03	1.12E−01	1.10E−01	1.79E−01	3.07E+00	4.29E+00	=
8	1.00E+02	1.00E+02	8.44E−14	1.00E+02	1.00E+02	2.53E−13	=
9	3.98E+02	4.01E+02	1.36E+00	4.15E+02	4.24E+02	5.11E+00	−
10	4.14E+02	4.14E+02	4.73E−04	3.99E+02	3.99E+02	6.47E−01	+
				2 + 2 −			

Table 19. DISH and jDE100 on CEC2020 in 5*D*.

f	DISH			jDE100			j2020		
	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std
1	2.60E−09	7.84E−09	1.87E−09	1.19E+05	9.94E+05	1.05E+06	0.00E+00	0.00E+00	0.00E+00
2	4.32E−09	3.99E−01	1.22E+00	1.46E+02	3.26E+02	8.04E+01	1.91E−04	3.23E+00	3.74E+00
3	6.13E−01	5.11E+00	8.63E−01	1.07E+01	2.00E+01	4.14E+00	0.00E+00	3.42E+00	2.33E+00
4	2.90E−09	6.82E−02	4.43E−02	2.69E−01	1.32E+00	4.48E−01	0.00E+00	7.68E−02	6.40E−02
5	1.83E−09	6.24E−02	1.90E−01	1.97E+01	5.80E+01	2.42E+01	0.00E+00	1.37E−01	2.86E−01
6	2.01E−09	6.94E−09	2.32E−09	4.19E−01	1.47E+00	5.28E−01	–	–	–
7	1.49E−09	5.96E−09	2.57E−09	1.95E+00	2.69E+01	2.96E+01	–	–	–
8	5.76E−09	3.35E+00	1.83E+01	4.06E+00	2.18E+01	9.07E+00	0.00E+00	6.28E−01	2.39E+00
9	1.00E+02	1.07E+02	2.54E+01	1.04E+02	1.23E+02	1.07E+01	0.00E+00	2.05E+01	3.75E+01
10	3.00E+02	3.44E+02	1.20E+01	1.89E+02	3.38E+02	3.24E+01	0.00E+00	1.26E+02	9.03E+01

Table 20. DISH and jDE100 on CEC2020 in 10*D*.

f	DISH			jDE100			j2020		
	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std
1	4.26E−09	8.83E−09	1.17E−09	5.88E+07	2.80E+08	1.48E+08	0.00E+00	0.00E+00	0.00E+00
2	6.25E−02	5.26E+00	3.92E+00	7.66E+02	1.06E+03	1.13E+02	0.00E+00	6.79E−01	1.16E+00
3	1.07E+01	1.19E+01	5.71E−01	6.45E+01	8.68E+01	1.23E+01	0.00E+00	8.06E+00	3.88E+00
4	1.28E−01	1.64E−01	2.53E−02	6.08E+00	1.01E+01	2.35E+00	0.00E+00	1.09E−01	9.04E−02
5	4.96E−09	2.43E−01	1.65E−01	4.28E+03	1.38E+04	6.69E+03	0.00E+00	3.02E−01	3.13E−01
6	1.97E−02	1.61E−01	1.28E−01	3.72E+01	1.15E+02	3.93E+01	2.91E−02	4.78E−01	2.49E−01
7	1.14E−07	3.31E−02	1.09E−01	5.59E+02	2.35E+03	1.37E+03	3.10E−07	6.73E−02	1.25E−01
8	1.00E+02	1.00E+02	0.00E+00	7.55E+01	1.35E+02	2.19E+01	0.00E+00	1.54E+00	4.00E+00
9	1.00E+02	2.67E+02	1.02E+02	1.63E+02	2.24E+02	2.24E+01	0.00E+00	8.00E+01	4.07E+01
10	3.98E+02	4.07E+02	1.84E+01	4.49E+02	4.73E+02	1.26E+01	1.00E+02	1.40E+02	8.12E+01

Table 21. DISH and jDE100 on CEC2020 in 15*D*.

f	DISH			jDE100			j2020		
	Best	**Mean**	**Std**	**Best**	**Mean**	**Std**	**Best**	**Mean**	**Std**
1	4.42E−09	8.41E−09	1.42E−09	6.48E+08	1.38E+09	5.36E+08	0.00E+00	0.00E+00	0.00E+00
2	1.67E−01	2.19E+01	4.02E+01	1.49E+03	2.02E+03	2.22E+02	0.00E+00	5.72E−02	4.32E−02
3	1.56E+01	1.67E+01	5.06E−01	1.41E+02	1.84E+02	2.41E+01	0.00E+00	6.78E+00	7.82E+00
4	1.78E−01	2.60E−01	3.93E−02	1.75E+01	7.69E+01	6.76E+01	0.00E+00	1.99E−01	7.47E−02
5	1.56E−01	2.54E+00	1.17E+00	3.17E+04	2.06E+05	9.67E+04	0.00E+00	7.58E+00	7.69E+00
6	2.07E−02	2.47E−01	2.06E−01	1.70E+02	3.36E+02	7.58E+01	1.65E−03	8.45E−01	2.09E+00
7	4.38E−01	7.59E−01	1.89E−01	1.51E+04	6.11E+04	3.24E+04	6.81E−02	9.83E−01	2.03E+00
8	1.00E+02	1.00E+02	0.00E+00	1.81E+02	2.82E+02	6.30E+01	0.00E+00	9.49E+00	2.74E+01
9	3.00E+02	3.84E+02	1.88E+01	2.98E+02	4.27E+02	4.89E+01	1.00E+02	1.23E+02	5.68E+01
10	4.00E+02	4.00E+02	0.00E+00	7.07E+02	8.67E+02	8.83E+01	1.00E+02	3.90E+02	5.48E+01

Table 22. DISH and jDE100 on CEC2020 in 20*D*.

f	DISH			jDE100			j2020		
	Best	**Mean**	**Std**	**Best**	**Mean**	**Std**	**Best**	**Mean**	**Std**
1	6.06E−09	8.44E−09	9.07E−10	1.76E+09	4.18E+09	1.38E+09	0.00E+00	0.00E+00	0.00E+00
2	6.25E−02	1.50E+00	1.69E+00	2.59E+03	3.11E+03	2.27E+02	0.00E+00	2.60E−02	2.47E−02
3	2.06E+01	2.16E+01	4.69E−01	2.29E+02	3.05E+02	3.33E+01	0.00E+00	1.44E+01	9.29E+00
4	2.56E−01	3.60E−01	4.24E−02	6.79E+01	4.38E+02	3.73E+02	2.98E−02	1.80E−01	7.84E−02
5	2.08E−01	6.77E+00	3.21E+00	3.58E+05	6.91E+05	1.88E+05	3.12E−01	7.78E+01	5.75E+01
6	3.67E−02	6.97E−01	4.64E−01	3.96E+02	6.86E+02	1.33E+02	6.84E−02	1.91E−01	1.01E−01
7	1.39E−02	1.17E−01	1.04E−01	2.25E+04	1.48E+05	5.43E+04	1.95E−02	1.98E+00	4.02E+00
8	1.00E+02	1.00E+02	2.23E−13	4.53E+02	7.18E+02	1.67E+02	0.00E+00	9.27E+01	2.21E+01
9	3.96E+02	4.01E+02	1.86E+00	5.04E+02	5.89E+02	3.25E+01	1.00E+02	3.39E+02	1.28E+02
10	4.14E+02	4.14E+02	5.12E−04	5.56E+02	8.34E+02	1.46E+02	1.00E+02	3.39E+02	1.28E+02

Convergence diagrams are shown in Figures 1–12. Figures 1–4 shows the convergence curves of SHADE and Tb-SHADE, respectively, for some test functions in 5*D*, 10*D*, 15*D*, and 20*D*, Figures 5–8 shows those of L-SHADE and TbL-SHADE for some test functions in 5*D*, 10*D*, 15*D*, and 20*D*. and Figures 9–12 shows those of the jSO and Tb-jSO, respectively, for some test functions in 5*D*, 10*D*, 15*D* and 20*D*. It is apparent that the red line of the turning-based mutation version of the algorithm was often slower to converge but attained better objective function values.

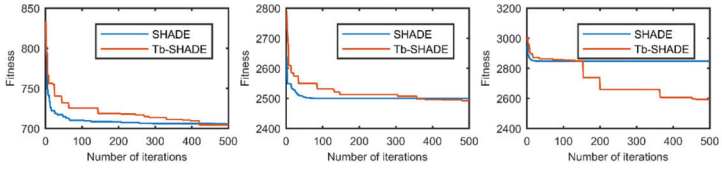


Figure 1. The selected average convergence of SHADE and Tb-SHADE on CEC2020 in 5*D* is compared. From left to right f3, f9 and f10.

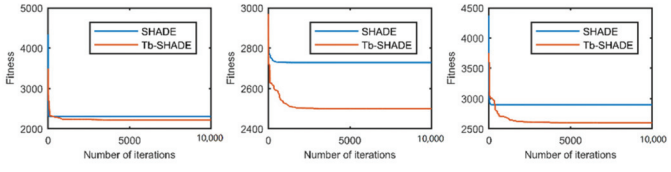


Figure 2. The selected average convergence of SHADE and Tb-SHADE is compared on CEC2020 in 10*D*. From left to right f8, f9 and f10.

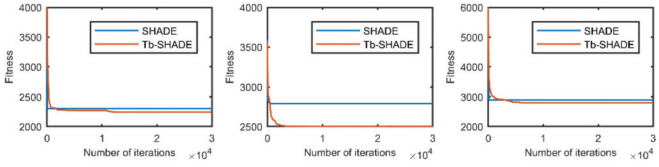


Figure 3. The selected average convergence of SHADE and Tb-SHADE is compared on CEC2020 in 15*D*. From left to right f8, f9 and f10.

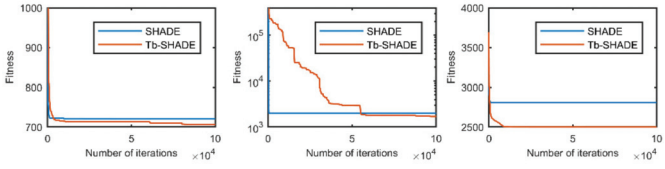


Figure 4. The selected average convergence of SHADE and Tb-SHADE is compared on CEC2020 in 20*D*. From left to right f3, f5 and f9.

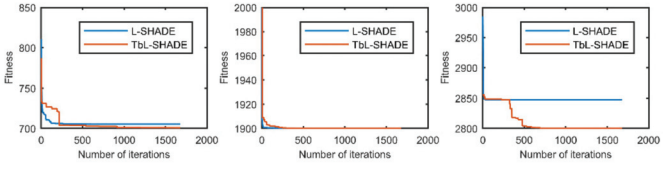


Figure 5. The selected average convergence of L-SHADE and TbL-SHADE is compared on CEC2020 in 5*D*. From left to right f3, f4 and f10.

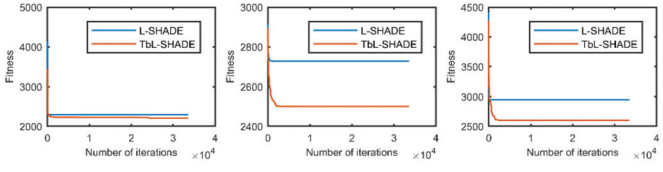


Figure 6. The selected average convergence of L-SHADE and TbL-SHADE is compared on CEC2020 in 10*D*. From left to right f8, f9 and f10.

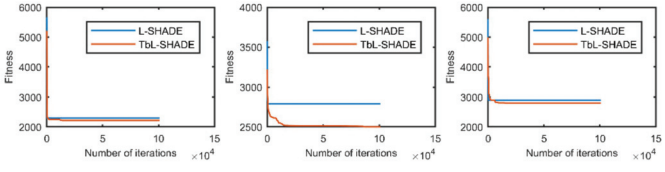


Figure 7. The selected average convergence of L-SHADE and TbL-SHADE is compared on CEC2020 in 15*D*. From left to right f8, f9 and f10.

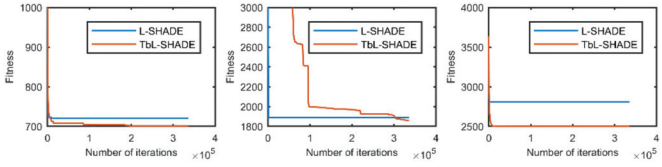


Figure 8. The selected average convergence of L-SHADE and TbL-SHADE is compared on CEC2020 in 20*D*. From left to right f3, f5 and f9.

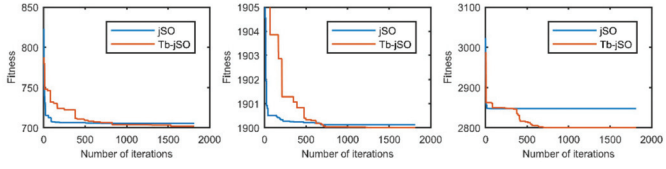


Figure 9. The selected average convergence of jSO and Tb-jSO is compared on CEC2020 in 5*D*. From left to right f3, f4 and f10.

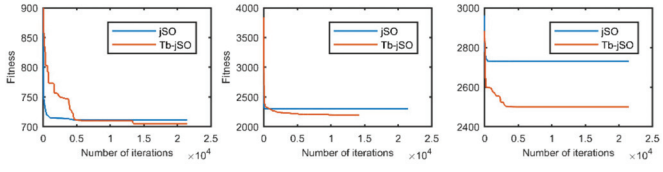


Figure 10. The selected average convergence of jSO and Tb-jSO is compared on CEC2020 in 10*D*. From left to right f3, f8 and f9.

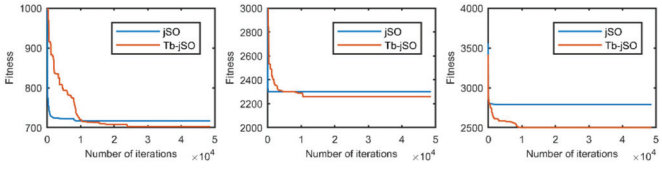


Figure 11. The selected average convergence of jSO and Tb-jSO is compared on CEC2020 in 15*D*. From left to right f3, f8 and f9.

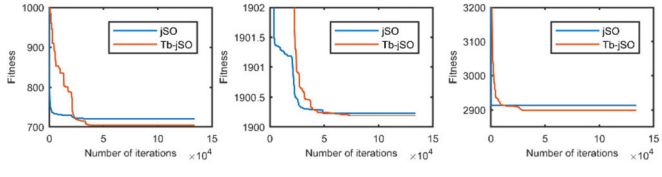


Figure 12. The selected average convergence of jSO and Tb-jSO is compared on CEC2020 in 20*D*. From left to right f3, f4 and f10.

Tables 23–34 shows the number of runs (#runs) of population aggregation, the average generation (Mean CO) of the first cluster during these runs, and the average population diversity (Mean PD) of these generations.

Table 23. Clustering and population diversity of SHADE and Tb-SHADE on the CEC2020 in 5*D*.

f	SHADE			Tb-SHADE		
	#runs	Mean CO	Mean PD	#runs	Mean CO	Mean PD
1	30	3.04E+01	3.74E+01	30	2.29E+02	4.84E+01
2	12	3.97E+02	7.29E+01	0	–	–
3	12	4.01E+02	1.20E+01	0	–	–
4	13	2.83E+02	1.16E+01	0	–	–
5	30	9.24E+01	3.69E+01	30	4.40E+02	3.66E+01
6	30	1.13E+02	3.29E+01	30	4.61E+02	2.28E+01
7	30	7.66E+01	4.57E+01	30	3.63E+02	4.41E+01
8	30	6.46E+01	2.84E+01	29	4.42E+02	2.52E+01
9	30	7.91E+01	6.31E+01	30	4.13E+02	3.61E+01
10	30	3.57E+01	1.30E+01	30	3.87E+02	3.03E+01

Table 24. Clustering and population diversity of SHADE and Tb-SHADE on the CEC2020 in 10*D*.

f	SHADE			Tb-SHADE		
	#runs	Mean CO	Mean PD	#runs	Mean CO	Mean PD
1	30	5.84E+01	1.88E+01	30	6.57E+02	8.30E+01
2	0	–	–	0	–	–
3	3	7.94E+03	1.35E+01	0	–	–
4	0	–	–	0	–	–
5	30	7.14E+02	3.92E+01	16	8.18E+03	8.16E+01
6	0	–	–	3	8.76E+03	8.21E+01
7	30	8.78E+02	2.36E+01	14	9.54E+03	3.66E+01
8	30	5.06E+01	1.48E+01	30	8.97E+02	1.39E+02
9	2	5.12E+03	3.09E+01	23	2.41E+03	1.12E+02
10	30	1.03E+02	2.37E+01	29	2.72E+03	7.69E+01

Table 25. Clustering and population diversity of SHADE and Tb-SHADE on the CEC2020 in 15*D*.

f	SHADE			Tb-SHADE		
	#runs	Mean CO	Mean PD	#runs	Mean CO	Mean PD
1	30	7.08E+01	1.37E+01	30	8.93E+02	1.05E+02
2	0	–	–	0	–	–
3	30	1.65E+04	9.53E+00	0	–	–
4	0	–	–	0	–	–
5	29	8.34E+02	3.33E+01	0	–	–
6	3	1.44E+04	6.27E+01	0	–	–
7	30	6.96E+02	1.56E+01	6	2.72E+04	6.18E+01
8	30	6.59E+01	1.12E+01	30	1.10E+03	2.16E+02
9	25	1.52E+04	3.12E+01	2	8.86E+03	9.17E+01
10	30	1.12E+02	6.72E+00	30	3.22E+03	1.13E+02

Table 26. Clustering and population diversity of SHADE and Tb-SHADE on the CEC2020 in 20*D*.

f	SHADE			Tb-SHADE		
	#runs	Mean CO	Mean PD	#runs	Mean CO	Mean PD
1	30	7.64E+01	1.19E+01	30	1.26E+03	8.92E+01
2	4	9.06E+04	7.74E+01	0	–	–
3	30	3.34E+04	9.12E+00	0	–	–
4	0	–	–	0	–	–
5	30	5.46E+02	1.86E+01	12	8.62E+04	1.27E+02
6	0	–	–	0	-	-
7	30	9.20E+02	2.59E+01	4	8.96E+04	4.63E+01
8	30	8.56E+01	1.06E+01	30	1.72E+03	2.82E+02
9	0	–	–	0	-	-
10	30	1.02E+02	6.21E+00	30	2.63E+03	1.26E+02

Table 27. Clustering and population diversity of L-SHADE and TbL-SHADE on the CEC2020 in 5*D*.

f	L-SHADE			TbL-SHADE		
	#runs	Mean CO	Mean PD	#runs	Mean CO	Mean PD
1	30	3.00E+01	3.75E+01	30	2.53E+02	4.85E+01
2	18	8.71E+02	3.20E+01	6	1.38E+03	1.66E+01
3	30	6.29E+02	7.01E+00	1	1.21E+03	6.85E+00
4	10	4.65E+02	1.06E+01	0	–	–
5	30	8.75E+01	3.42E+01	30	1.10E+03	2.17E+01
6	30	1.14E+02	3.01E+01	30	7.34E+02	2.30E+01
7	30	7.66E+01	4.29E+01	30	4.85E+02	4.13E+01
8	30	6.47E+01	2.08E+01	30	7.59E+02	1.97E+01
9	30	7.11E+01	6.15E+01	30	3.90E+02	2.77E+01
10	30	3.52E+01	1.07E+01	30	3.71E+02	2.93E+01

Table 28. Clustering and population diversity of L-SHADE and TbL-SHADE on the CEC2020 in 10*D*.

f	L-SHADE			TbL-SHADE		
	#runs	Mean CO	Mean PD	#runs	Mean CO	Mean PD
1	30	5.96E+01	2.01E+01	30	6.61E+02	8.21E+01
2	10	2.88E+04	2.98E+01	9	3.12E+04	2.12E+01
3	20	2.65E+04	4.36E+00	2	3.19E+04	3.52E+00
4	1	2.86E+04	8.35E+00	2	3.21E+04	1.97E+00
5	30	7.31E+02	4.15E+01	15	2.89E+04	1.58E+01
6	6	2.31E+04	1.10E+01	0	–	–
7	30	9.46E+02	2.34E+01	30	2.30E+04	1.77E+01
8	30	5.75E+01	1.51E+01	30	8.86E+02	1.26E+02
9	10	1.63E+04	5.77E+01	26	6.57E+03	1.18E+02
10	30	1.08E+02	2.70E+01	28	4.24E+03	7.02E+01

Table 29. Clustering and population diversity of L-SHADE and TbL-SHADE on the CEC2020 in 15*D*.

f	L-SHADE			TbL-SHADE		
	#runs	Mean CO	Mean PD	#runs	Mean CO	Mean PD
1	30	7.06E+01	1.45E+01	30	8.80E+02	1.05E+02
2	30	6.19E+04	4.25E+01	30	7.69E+04	3.52E+01
3	30	1.84E+04	7.97E+00	24	8.51E+04	2.14E+01
4	4	9.42E+04	6.03E+00	3	9.80E+04	1.64E+00
5	28	8.35E+02	3.80E+01	2	9.80E+04	1.21E+01
6	18	7.04E+04	1.09E+01	17	9.20E+04	2.24E+01
7	30	6.98E+02	1.54E+01	0	–	–
8	30	6.60E+01	1.15E+01	30	1.10E+03	2.16E+02
9	30	2.00E+04	2.82E+01	14	7.38E+04	1.05E+02
10	30	1.14E+02	6.87E+00	30	3.13E+03	1.08E+02

Table 30. Clustering and population diversity of L-SHADE and TbL-SHADE on the CEC2020 in 20*D*.

f	L-SHADE			TbL-SHADE		
	#runs	Mean CO	Mean PD	#runs	Mean CO	Mean PD
1	30	7.64E+01	1.22E+01	30	1.26E+03	8.95E+01
2	26	9.06E+04	6.00E+01	30	1.79E+05	6.37E+01
3	30	3.61E+04	8.72E+00	30	2.21E+05	4.83E+01
4	17	3.00E+05	1.14E+01	9	3.03E+05	6.49E+00
5	30	5.46E+02	1.63E+01	19	2.66E+05	7.85E+01
6	17	2.59E+05	1.75E+01	11	2.94E+05	1.89E+01
7	30	8.89E+02	1.92E+01	7	3.24E+05	1.18E+01
8	30	8.64E+01	8.66E+00	30	1.72E+03	2.80E+02
9	19	2.37E+05	3.59E+01	26	2.19E+05	1.11E+02
10	30	1.04E+02	6.78E+00	30	2.52E+03	1.26E+02

Table 31. Clustering and population diversity of jSO and Tb-jSO on the CEC2020 in 5*D*.

f	jSO			Tb- jSO		
	#runs	Mean CO	Mean PD	#runs	Mean CO	Mean PD
1	30	3.17E+01	4.57E+01	30	3.17E+02	4.18E+01
2	29	9.61E+02	4.16E+01	26	1.22E+03	4.92E+01
3	30	9.12E+02	9.36E+00	19	1.43E+03	2.33E+01
4	29	1.08E+03	8.68E+00	22	1.44E+03	1.39E+01
5	30	1.17E+02	3.36E+01	30	9.94E+02	2.81E+01
6	30	1.57E+02	3.68E+01	30	9.66E+02	1.54E+01
7	30	1.06E+02	4.38E+01	29	6.79E+02	3.89E+01
8	30	7.21E+01	1.74E+01	30	7.20E+02	2.13E+01
9	30	5.49E+01	6.52E+01	30	3.94E+02	2.70E+01
10	30	3.25E+01	9.66E+00	30	3.90E+02	2.97E+01

Table 32. Clustering and population diversity of jSO and Tb-jSO on the CEC2020 in 10*D*.

f	jSO			Tb-jSO		
	#runs	Mean CO	Mean PD	#runs	Mean CO	Mean PD
1	30	5.88E+01	3.93E+01	30	2.29E+03	7.97E+01
2	30	6.11E+03	1.12E+02	30	6.86E+03	1.65E+02
3	30	6.17E+03	1.48E+01	30	1.04E+04	4.81E+01
4	30	8.55E+03	1.28E+01	30	1.34E+04	2.44E+01
5	30	4.41E+03	3.63E+01	30	8.97E+03	5.36E+01
6	30	5.54E+03	2.38E+01	30	1.07E+04	5.42E+01
7	30	1.44E+03	3.57E+01	30	9.69E+03	3.94E+01
8	30	5.52E+01	9.59E+00	30	3.40E+03	5.53E+01
9	30	4.98E+03	4.19E+01	30	3.73E+03	4.66E+01
10	30	8.33E+01	2.45E+01	30	3.07E+03	6.99E+01

Table 33. Clustering and population diversity of jSO and Tb-jSO on the CEC2020 in 15*D*.

f	jSO			Tb- jSO		
	#runs	Mean CO	Mean PD	#runs	Mean CO	Mean PD
1	30	8.09E+01	4.39E+01	30	5.34E+03	1.02E+02
2	30	1.39E+04	1.21E+02	30	1.54E+04	1.66E+02
3	30	1.20E+04	1.37E+01	30	2.02E+04	5.26E+01
4	30	1.69E+04	1.57E+01	30	2.99E+04	2.84E+01
5	30	1.17E+04	4.95E+01	30	1.31E+04	1.24E+02
6	30	1.79E+04	2.23E+01	30	2.41E+04	5.03E+01
7	30	2.53E+03	3.56E+01	30	2.26E+04	3.76E+01
8	30	7.50E+01	7.39E+00	30	3.53E+03	4.65E+01
9	30	1.07E+04	3.03E+01	30	1.83E+04	4.07E+01
10	30	9.39E+01	6.56E+00	30	8.49E+03	8.97E+01

Table 34. Clustering and population diversity of jSO and Tb-jSO on the CEC2020 in 20*D*.

f	jSO			Tb-jSO		
	#runs	Mean CO	Mean PD	#runs	Mean CO	Mean PD
1	30	9.32E+01	3.60E+01	30	9.38E+03	9.36E+01
2	30	3.15E+04	1.17E+02	30	3.48E+04	1.69E+02
3	30	2.91E+04	1.37E+01	30	4.65E+04	6.03E+01
4	30	3.86E+04	1.80E+01	30	7.28E+04	3.40E+01
5	30	2.94E+04	6.35E+01	30	3.25E+04	1.28E+02
6	30	3.81E+04	2.77E+01	30	4.80E+04	9.32E+01
7	30	3.06E+04	1.92E+01	30	4.83E+04	6.16E+01
8	30	9.09E+01	7.40E+00	30	7.11E+03	4.78E+01
9	30	2.84E+04	4.81E+01	30	3.68E+04	1.08E+02
10	30	9.94E+01	6.62E+00	30	1.08E+04	1.27E+02

The rankings of the Friedman test [52] were obtained by using the average value (Mean) of each algorithm on all 10 test functions in Tables 7–22, and are shown in Tables 35–38. The related statistical

values of the Friedman test are shown in Table 39. If the chi-square statistic was greater than the critical value, the null hypothesis was rejected. p represents the probability of the null hypothesis obtaining. The null hypothesis here was that there is no significant difference in performance among the nine algorithms considered here on CEC2020.

Table 35. The Friedman ranks of comparative algorithms on CEC2020 in 5D.

Rank	Name	F-Rank
0	TbL-SHADE	3.2
1	Tb-jSO	3.55
2	L-SHADE	3.8
3	jSO	4.05
4	j2020	4.95
5	DISH	5.05
6	SHADE	5.2
7	Tb-SHADE	6.6
8	jDE100	8.6

Table 36. The Friedman ranks of comparative algorithms on CEC2020 in 10D.

Rank	Name	F-Rank
0	j2020	3.2
1	Tb-jSO	3.6
2	TbL-SHADE	3.9
3	DISH	4.9
4	jSO	4.95
5	Tb-SHADE	5.05
6	L-SHADE	5.05
7	SHADE	5.75
8	jDE100	8.6

Table 37. The Friedman ranks of comparative algorithms on CEC2020 in 15D.

Rank	Name	F-Rank
0	j2020	3.15
1	TbL-SHADE	3.45
2	Tb-jSO	3.45
3	Tb-SHADE	4.35
4	DISH	4.7
5	jSO	5.2
6	L-SHADE	5.6
7	SHADE	6.1
8	jDE100	9

Table 38. The Friedman ranks of comparative algorithms on CEC2020 in 20D.

Rank	Name	F-Rank
0	j2020	2.35
1	TbL-SHADE	4.05
2	Tb-jSO	4.3
3	DISH	4.8
4	jSO	4.9
5	Tb-SHADE	5
6	L-SHADE	5.1
7	SHADE	5.5
8	jDE100	9

Table 39. Related statistical values obtained of Friedman test for $\alpha = 0.05$.

D	Chi-sq′	Prob > Chi-sq′ (*p*)	Critical Value·
5	34.25414365	3.65E−05	15.51
10	28.83468835	3.39E−04	15.51
15	38.8213628	5.31E−06	15.51
20	37.00654818	1.15E−05	15.51

6. Results and Discussion

The results on the CEC2020 benchmark sets are first discussed. As shown in Tables 7–18, the scores were two improvements against two instances of worsening (5*D*), four improvements and two instances of worsening (10*D*), five improvements and two instances of worsening (15*D*), and four improvements no instances of worsening (20*D*) in the case of SHADE; three improvements against zero instances of worsening (5*D*), four improvements and one worsening (10*D*), six improvements no worsening (15*D*), and five improvements and no worsening (20*D*) in the case of L-SHADE; and one improvement against no worsening (5*D*), four improvements and one worsening (10*D*), four improvements and one worsening (15*D*), and two improvements two instances of worsening (20*D*) in the case of jSO. In some test functions, the improved algorithm even escaped the local optimum and found the optimal value (if the error was smaller than 10^{-8}, the relevant value was considered optimal). Examples are f3 in Tables 10 and 13, and Table 14, f8 in Tables 9, 13 and 16, and Table 17, f9 in Table 12, and f10 in Table 11. In most cases, the improved version was clearly better than the original algorithm except for Tb-SHADE (5*D*) and Tb-jSO (20*D*).

According to the convergence curves in Figures 1–12, in most cases, the improved algorithm showed similar convergence to the original in the early stage of the optimization process, but it clearly maintained a longer exploration phase and achieved better values of the objective function in the middle and late stages; in a few cases (such as $f4$ in Figure 5), the improved algorithm had slower convergence but did not achieve a better objective function value than the original.

As the numerical analyses in Tables 23–34 show, in most cases, the improved algorithms exhibited fewer clusters (#runs), later clustering (mean CO), and higher population density (mean PD) than the original algorithm. But Tb-SHADE (5*D*) had a lower population density on $f6$–$f9$, as did TbL-SHADE (all dimensions) on $f2$–$f7$, where this might have been related to the linear decrease in the population size. Tb-jSO showed similar numbers of clusters in all dimensions and a lower population density on some test functions in 5*D*. Therefore, in most cases, the improved versions maintained the diversity of population and a longer exploration phase in the optimization process.

The significant improvements in Tables 7–18 and the clustering analysis in Tables 23 and 24 can be linked. The results in the former set of tables with the "+" symbol were always connected with the occurrence of later clustering, none at all, or fewer instances of clusters of 30 (for the last option, see, for example, column #runs in Tables 24–26, $f3$). Consequently, the improvement in the performance effected by the updated version was related to the maintenance of population diversity and a longer exploration phase.

According to the Friedman ranking in Tables 35–38, Tb-SHADE, TbL-SHADE, and Tb-jSO were clearly better than the original algorithms and the advanced DISH and jDE100 in 10*D*, 15*D*, and 20*D*. But Tb-SHADE did not perform as well as SHADE in 5*D* and did not perform as well as DISH in 5*D*, 10*D* and 20*D*. In addition, the j2020 algorithm delivered the best performance and ranked first in 10*D*, 15*D* and 20*D* and one of the improved versions, TbL-SHADE, only delivered the best performance and ranked first in 5*D*. And jDE100 (winner of CEC2019), which ranks last in Tables 35–38, did not seem suitable for CEC2020. Table 39 shows that the null hypothesis was rejected in all dimensions, and thus the Friedman ranking was correct. All in all, the three improved algorithms obtained good optimization results in contrast to the original algorithm as well as the advanced DISH and jDE100 algorithms but were slightly worse than the advanced j2020 algorithm.

7. Conclusions

In this paper, a relatively simple and direct method using turning-based mutation was proposed and tested on Single Objective Bound Constrained Numerical Optimization (CEC2020) benchmark sets in 5, 10, 15, and 20 dimensions against the SHADE, L-SHADE, and jSO algorithms. The basic thought of the proposed method is to change the direction of mutation under certain conditions to maintain the population diversity and a longer exploration phase. It can thus avoid premature convergence and escape the local optimum to get better optimization results. The results of experiments showed that this method is effective on CEC2020 benchmark sets in 10, 15, and 20 dimensions. The strong point of the proposed method is that it can be applied to variants of SHADE easily. A disadvantage is that it increases the time complexity and its effectiveness lacks theoretical proof. Our future research in the area will focus on further experiments, and on applying the proposed method to more algorithms. For example, the improved method may be useful for some practical problems featuring constraints.

Author Contributions: Conceptualization, H.K.; methodology, H.K.; project administration, L.J. and Y.S.; software, X.S.; validation, X.S. and Q.C.; visualization, L.J. and Q.C.; formal analysis, H.K.; investigation, Q.C.; resources, Y.S.; data curation, L.J.; writing—original draft preparation, L.J.; writing—review and editing, H.K. and X.S.; supervision: X.S.; funding acquisition: Y.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China, grant number 61663046, 61876166. This research was funded by Open Foundation of Key Laboratory of Software Engineering of Yunnan Province, grant number 2015SE204.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Storn, R.; Price, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
2. Eltaeib, T.; Mahmood, A. Differential evolution: A survey and analysis. *Appl. Sci.* **2018**, *8*, 1945. [CrossRef]
3. Arafa, M.; Sallam, E.A.; Fahmy, M. An enhanced differential evolution optimization algorithm. In Proceedings of the 2014 Fourth International Conference on Digital Information and Communication Technology and its Applications (DICTAP), Bangkok, Thailand, 6–8 May 2014; pp. 216–225.
4. Awad, N.H.; Ali, M.Z.; Suganthan, P.N. Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June 2017; pp. 372–379.
5. Bujok, P.; Tvrdík, J. Adaptive differential evolution: SHADE with competing crossover strategies. In Proceedings of the International Conference on Artificial Intelligence and Soft Computing, Zakopane, Poland, 14–18 June 2015; pp. 329–339.
6. Bujok, P.; Tvrdík, J.; Poláková, R. Evaluating the performance of shade with competing strategies on CEC 2014 single-parameter test suite. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 5002–5009.
7. Liu, X.-F.; Zhan, Z.-H.; Zhang, J. Dichotomy guided based parameter adaptation for differential evolution. In Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, Madrid, Spain, 11–15 July 2015; pp. 289–296.
8. Liu, Z.-G.; Ji, X.-H.; Yang, Y. Hierarchical differential evolution algorithm combined with multi-cross operation. *Expert Syst. Appl.* **2019**, *130*, 276–292. [CrossRef]
9. Mohamed, A.W.; Hadi, A.A.; Fattouh, A.M.; Jambi, K.M. LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June 2017; pp. 145–152.
10. Poláková, R.; Tvrdík, J.; Bujok, P. L-SHADE with competing strategies applied to CEC2015 learning-based test suite. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 4790–4796.

11. Poláková, R.; Tvrdík, J.; Bujok, P. Evaluating the performance of L-SHADE with competing strategies on CEC2014 single parameter-operator test suite. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 1181–1187.
12. Sallam, K.M.; Sarker, R.A.; Essam, D.L.; Elsayed, S.M. Neurodynamic differential evolution algorithm and solving CEC2015 competition problems. In Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC), Sendai, Japan, 25–28 May 2015; pp. 1033–1040.
13. Viktorin, A.; Pluhacek, M.; Senkerik, R. Network based linear population size reduction in SHADE. In Proceedings of the 2016 International Conference on Intelligent Networking and Collaborative Systems (INCoS), Ostrawva, Czech Republic, 7–9 September 2016; pp. 86–93.
14. Viktorin, A.; Pluhacek, M.; Senkerik, R. Success-history based adaptive differential evolution algorithm with multi-chaotic framework for parent selection performance on CEC2014 benchmark set. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 4797–4803.
15. Viktorin, A.; Senkerik, R.; Pluhacek, M.; Kadavy, T. Distance vs. Improvement Based Parameter Adaptation in SHADE. In *Artificial Intelligence and Algorithms in Intelligent Systems*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 455–464.
16. Viktorin, A.; Senkerik, R.; Pluhacek, M.; Kadavy, T.; Zamuda, A. Distance based parameter adaptation for differential evolution. In Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, USA, 27 November–1 December 2017; pp. 1–7.
17. Zhao, F.; He, X.; Yang, G.; Ma, W.; Zhang, C.; Song, H. A hybrid iterated local search algorithm with adaptive perturbation mechanism by success-history based parameter adaptation for differential evolution (SHADE). *J. Eng. Optim.* **2020**, *52*, 367–383. [CrossRef]
18. Al-Dabbagh, R.D.; Neri, F.; Idris, N.; Baba, M.S. Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy. *Swarm Evol. Comput.* **2018**, *43*, 284–311. [CrossRef]
19. Tanabe, R.; Fukunaga, A. Success-history based parameter adaptation for differential evolution. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; pp. 71–78.
20. Zhang, J.; Sanderson, A.C. JADE: Adaptive differential evolution with optional external archive. *IEEE Trans. Evol. Comput.* **2009**, *13*, 945–958. [CrossRef]
21. Tanabe, R.; Fukunaga, A.S. Improving the search performance of SHADE using linear population size reduction. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014; pp. 1658–1665.
22. Guo, S.-M.; Tsai, J.S.-H.; Yang, C.-C.; Hsu, P.-H. A self-optimization approach for L-SHADE incorporated with eigenvector-based crossover and successful-parent-selecting framework on CEC 2015 benchmark set. In Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC), Sendai, Japan, 25–28 May 2015; pp. 1003–1010.
23. Awad, N.H.; Ali, M.Z.; Suganthan, P.N.; Reynolds, R.G. An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving CEC2014 benchmark problems. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 2958–2965.
24. Brest, J.; Maučec, M.S.; Bošković, B. Single objective real-parameter optimization: Algorithm jSO. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June 2017; pp. 1311–1318.
25. Piotrowski, A.P.; Napiorkowski, J.J. Step-by-step improvement of JADE and SHADE-based algorithms: Success or failure? *Swarm Evol. Comput.* **2018**, *43*, 88–108. [CrossRef]
26. Piotrowski, A.P. L-SHADE optimization algorithms with population-wide inertia. *Inf. Sci.* **2018**, *468*, 117–141. [CrossRef]
27. Stanovov, V.; Akhmedova, S.; Semenkin, E. LSHADE algorithm with rank-based selective pressure strategy for solving CEC 2017 benchmark problems. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
28. Brest, J.; Maučec, M.S.; Bošković, B. The 100-Digit Challenge: Algorithm jDE100. In Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 10–13 June 2019; pp. 19–26.
29. Brest, J.; Maučec, M.S.; Bošković, B. Differential Evolution Algorithm for Single Objective Bound-Constrained Optimization: Algorithm j2020. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; pp. 1–8.

30. Suganthan, P.N. Particle swarm optimiser with neighbourhood operator. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999; pp. 1958–1962.
31. Das, S.; Maity, S.; Qu, B.-Y.; Suganthan, P.N. Real-parameter evolutionary multimodal optimization—A survey of the state-of-the-art. *Swarm Evol. Comput.* **2011**, *1*, 71–88. [CrossRef]
32. Mezura-Montes, E.; Coello, C.A.C. Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm Evol. Comput.* **2011**, *1*, 173–194. [CrossRef]
33. Neri, F.; Cotta, C. Memetic algorithms and memetic computing optimization: A literature review. *Swarm Evol. Comput.* **2012**, *2*, 1–14. [CrossRef]
34. Neri, F.; Tirronen, V. Recent advances in differential evolution: A survey and experimental analysis. *Artif. Intell. Rev.* **2010**, *33*, 61–106. [CrossRef]
35. Zamuda, A.; Brest, J. Self-adaptive control parameters' randomization frequency and propagations in differential evolution. *Swarm Evol. Comput.* **2015**, *25*, 72–99. [CrossRef]
36. Zhou, A.; Qu, B.-Y.; Li, H.; Zhao, S.-Z.; Suganthan, P.N.; Zhang, Q. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm Evol. Comput.* **2011**, *1*, 32–49. [CrossRef]
37. Piotrowski, A.P. Review of differential evolution population size. *Swarm Evol. Comput.* **2017**, *32*, 1–24. [CrossRef]
38. Opara, K.R.; Arabas, J. Differential Evolution: A survey of theoretical analyses. *Swarm Evol. Comput.* **2019**, *44*, 546–558. [CrossRef]
39. Poikolainen, I.; Neri, F.; Caraffini, F. Cluster-based population initialization for differential evolution frameworks. *Inf. Sci.* **2015**, *297*, 216–235. [CrossRef]
40. Weber, M.; Neri, F.; Tirronen, V. A study on scale factor/crossover interaction in distributed differential evolution. *Artif. Intell. Rev.* **2013**, *39*, 195–224. [CrossRef]
41. Zaharie, D. Influence of crossover on the behavior of differential evolution algorithms. *Appl. Soft Comput.* **2009**, *9*, 1126–1138. [CrossRef]
42. Yue, C.; Price, K.; Suganthan, P.; Liang, J.; Ali, M.; Qu, B.; Awad, N.; Biswas, P. Problem definitions and evaluation criteria for the CEC 2020 special session and competition on single objective bound constrained numerical optimization. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020.
43. Piotrowski, A.P.; Napiorkowski, J.J. Some metaheuristics should be simplified. *Inf. Sci.* **2018**, *427*, 32–62. [CrossRef]
44. Viktorin, A.; Senkerik, R.; Pluhacek, M.; Kadavy, T.; Zamuda, A. Distance based parameter adaptation for success-history based differential evolution. *Swarm Evol. Comput.* **2019**, *50*, 100462. [CrossRef]
45. Caraffini, F.; Kononova, A.V.; Corne, D. Infeasibility and structural bias in differential evolution. *Inf. Sci.* **2019**, *496*, 161–179. [CrossRef]
46. Awad, N.; Ali, M.; Liang, J.; Qu, B.; Suganthan, P.; Definitions, P. Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June 2017.
47. Brest, J.; Maučec, M.S.; Bošković, B. iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 1188–1195.
48. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M. Opposition-based differential evolution. *IEEE Trans. Evol. Comput.* **2008**, *12*, 64–79. [CrossRef]
49. Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD-96 Proceedings*; AAAI: Menlo Park, CA, USA, 1996; pp. 226–231.
50. Deza, M.M.; Deza, E. Encyclopedia of distances. In *Encyclopedia of Distances*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 1–583.
51. Poláková, R.; Tvrdík, J.; Bujok, P.; Matoušek, R. Population-size adaptation through diversity-control mechanism for differential evolution. In Proceedings of the MENDEL, 22th International Conference on Soft Computing, Brno, Czech Republic, 8–10 June 2016; pp. 49–56.
52. Demšar, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.

mathematics

MDPI

Article

Opposition-Based Ant Colony Optimization Algorithm for the Traveling Salesman Problem

Zhaojun Zhang [1,†], Zhaoxiong Xu [1,†], Shengyang Luan [1,†], Xuanyu Li [1] and Yifei Sun [2,*]

[1] School of Electrical Engineering and Automation, Jiangsu Normal University, Xuzhou 221116, China; zzj921@163.com (Z.Z.); xzx94819@163.com (Z.X.); luan@jsnu.edu.cn (S.L.); lixuanyu5705@163.com (X.L.)

[2] School of Computer Science & School of Physics and Information Technology, Shaanxi Normal University, Xi'an 710119, China

* Correspondence: yifeis@snnu.edu.cn; Tel.: +86-157-0521-5977

† These authors contributed equally to this work.

Received: 18 August 2020; Accepted: 22 September 2020 ; Published: 24 September 2020

Abstract: Opposition-based learning (OBL) has been widely used to improve many swarm intelligent optimization (SI) algorithms for continuous problems during the past few decades. When the SI optimization algorithms apply OBL to solve discrete problems, the construction and utilization of the opposite solution is the key issue. Ant colony optimization (ACO) generally used to solve combinatorial optimization problems is a kind of classical SI optimization algorithm. Opposition-based ACO which is combined in OBL is proposed to solve the symmetric traveling salesman problem (TSP) in this paper. Two strategies for constructing opposite path by OBL based on solution characteristics of TSP are also proposed. Then, in order to use information of opposite path to improve the performance of ACO, three different strategies, direction, indirection, and random methods, mentioned for pheromone update rules are discussed individually. According to the construction of the inverse solution and the way of using it in pheromone updating, three kinds of improved ant colony algorithms are proposed. To verify the feasibility and effectiveness of strategies, two kinds of ACO algorithms are employed to solve TSP instances. The results demonstrate that the performance of opposition-based ACO is better than that of ACO without OBL.

Keywords: opposition-based learning; ant colony optimization; opposite path; traveling salesman problems

1. Introduction

As an important branch of computational intelligence, swarm intelligence (SI) [1] provides a competitive solution for dealing with large-scale, nonlinear, and complex problems, and has become an important research direction of artificial intelligence. In the SI model, each individual constitutes an organic whole by simulating the behavior of natural biological groups. Although each individual is very simple, the group shows complex emergent behavior. In particular, it does not require prior knowledge of the problem and has the characteristics of parallelism, so it has significant advantages in dealing with problems that are difficult to solve by traditional optimization algorithms. With the deepening of research, more and more swarm intelligence algorithms have been proposed, such as ant colony optimization algorithm (ACO) [2], particle swarm optimization (PSO) [3], artificial bee colony algorithm (ABC) [4], firefly algorithm (FA) [5], cuckoo algorithm (CA) [6], krill herd algorithm [7], monarch butterfly optimization (MBO) [8], and moth search algorithm [9], etc.

ACO as one of the typical SI is first proposed by Macro Dorigo [2] based on the observation of group behaviors of ants in nature. During the process of food searching, ants will release pheromones in the path when they pass through. Pheromones can be detected by other ants and can affect their further path choices. Generally, the shorter the path is, the more intense the pheromones will be, which means the shortest path will be chosen with the highest probability. The pheromone in other

paths will disappear with time. Therefore, given enough time, the optimal path will have the most condensed pheromone. In this way, ants will find the shortest path from their nest to the food source in the end.

ACO has advantages in reasonable robustness, distributed parallel computing, and easy combination with other algorithms. It has been successfully applied in many fields, including traveling salesman problem (TSP) [10,11], satellite control resource scheduling problem [12], knapsack problem [13,14], vehicle routing problem [15,16], and continuous function optimization [17–19]. However, conventional ACO is still far from perfect due to issues like premature convergence and long search time [20].

Many scholars have made substantial contributions to improve ACO, mainly focusing on two perspectives, including model modification and algorithms combination. For example, in the line of model improvement, an ant colony system (ACS) [21] employs a pseudo-random proportional rule, which leads to faster convergence. In ACS, only the pheromone of the optimal path will be increased after each iteration. To prevent premature convergence caused by excessive pheromones concentration in some paths, the max-min ant system (MMAS) modifies AS with three main strategies for pheromone [22], including limitation, maximum initialization, and updating rules. To avoid the early planning of the blind search, an improved ACO algorithm by constructing the unequal allocation initial pheromones is proposed in [23]. Path selecting is based on the pseudo-random rule for state transition. The probability is decided by the number of iterations, and the optimal solution. Introducing a penalty function to the pheromone updating, a novel ACO algorithm is addressed in [24] to improve the solution accuracy.

Considering the other primary kind of modification to the original ACO, algorithm combination, several approaches are proposed as well. A multi-type ant system (MTAS) [25] is proposed combining ACS and MMAS, inheriting advantages from both of these algorithms. Combining particle swarm algorithm (PSO) with ACO, a new ant colony algorithm was proposed in [26] and named PS-ACO. PS-ACO employs pheromones updating rules of ACO and searches mechanisms of PSO simultaneously to keep the trade-off between the exploitation and exploration. A multi-objective evolutionary algorithm via decomposition is combined with ACO, an algorithm, termed MOEA/D-ACO [27], which proposes a series of single-objective optimization problems to solve multi-objective optimization problems. Executing ACO in combination with a genetic algorithm (GA), a new hybrid algorithm is proposed in [28]. Embedding GA into ACO, this method improves ACO in convergence speed and GA in searching ability.

Besides the above primary improvement strategies considering model modification and algorithm combination, approaches based on machine learning are also proposed in recent decades [29]. On the one hand, swarm intelligence can be used to solve the optimization problems in deep learning. In deep neural networks, for example, convolutional neural network (CNN), the optimization of hyperparameters is an NP hard problem. Using the SI method can solve this kind of problem better. PSO, CS, and FA were employed to properly select dropout parameters concerning CNN in [30]. The hybridized algorithm [31] based on original MBO with ABC and FAs was proposed to solve CNN hyperparameters optimization. On the other hand, we can learn from machine learning to improve performance of SI. For example, information feedback models are used to enhance the ability of algorithms [32–34]. In addition, opposition-based learning (OBL) [35], which was first proposed by Tizhoosh, is a famous algorithm. Its main idea is to calculate all the opposite solutions after current iteration, and then optimal solutions are selected among the generated solutions and their opposite solutions for the next round of iteration. OBL has been widely accepted in SI, including ABC [36], differential evolution (DE) [37–39], and PSO [40,41], leading to reasonable performances.

Since opposite solutions to continuous problems are convenient to construct, OBL has been used to solve continuous problems more commonly as above, compared with discrete problems. OBL is combined with ACS and applied to solve the TSP as an example for discrete problems in [42] to acquire the better solution. The solution construction phase and the pheromone updating phase of ACS are

the primary foci of this hybrid approach. Besides TSP, the graph coloring problem is also employed as a discrete optimization problem in [43], and an improved DE algorithm based on OBL is proposed, which introduces two different methods of opposition. In [44], a pretreatment step was added in the initial stage when the two-membered evolution strategy was used to solve the total rotation minimization problem. The opposite solutions generated by OBL is compared with the initial solutions randomly generated, and a better solution is selected for the subsequent optimization process.

Inspired by the idea of OBL, in this paper, a series of methods, focusing on the opposite solution construction and the pheromone updating rule, are proposed. Aiming to solve TSP, our proposed methods introduce OBL to ACO and enable ACO no longer limited to the local optimal solutions, avoid premature convergence, and improve its performances.

The rest of this paper is organized as follows. In Section 2, the background knowledge of ACO and OBL are briefly reviewed. In Section 3, the opposition-based extensions to ACO are presented. In Section 4, the effectiveness of the improvement is verified through experiments. Section 5 presents the conclusions of this paper.

2. Background

In this section, we will take AS as an example to introduce the main process of ACO algorithm. At the same time, some necessary explanations of OBL will also be given.

2.1. Ant System

TSP can be described as finding the shortest route for a salesman who needs to visit each city at least once and no more than once [45]. TSP is a classical combination optimization problem which is employed to test ACO algorithms, and, therefore, TSP is used here as an example as well. The TSP includes symmetric TSP and asymmetric TSP. We only discuss symmetric TSP in this paper.

There are two primary steps in the AS algorithm, path construction, and pheromone updating [2]. During the first step, a solution is established according to the random proportion rule, and it can be described in detail as follows.

In the beginning, m ants are randomly assigned to n cities. At the t-th iteration, the probability, called the state transition probability, for the k-th ant to travel from the city i to j is

$$p_{ij}^{k}(t) = \begin{cases} \frac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}(t)\right]^{\beta}}{\sum_{s \in J_k(i)}\left[\tau_{is}(t)\right]^{\alpha}\left[\eta_{is}(t)\right]^{\beta}}, & \text{if } j \in J_k(i) \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where τ_{ij} is the pheromone trail and η_{ij} is the heuristic information, accordingly, while α and β are parameters deciding their relative influences, respectively. Generally, $\eta_{ij} = 1/d_{ij}$ and d_{ij} is the distance of the path (i, j). $J_k(i)$ is the feasible neighborhood of k-th ant at the i-th city.

When all the ants finish touring around each city, pheromone updating is as follows:

$$\tau_{ij}(t+1) = (1-\rho)\,\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{2}$$

where $\rho\ (0 < \rho \leq 1)$ is the evaporation rate, $\Delta\tau_{ij}^{k}$ represents the extra pheromone left in the path (i, j) by the k-th ant. $\Delta\tau_{ij}^{k}$ could be decided through

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L_k}, & \text{if ant } k \text{ passes the path } (i,j) \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where Q is the pheromone enhancement coefficient and L_k is the total path length for the k-th ant.

2.2. Opposition-Based Learning

In the continuous domain, OBL is employed to evaluate the current solutions and their opposite solutions. Among these solutions, optimal ones are selected to boost the searchability [46]. Relative definitions are given as follows.

Definition 1. *Let $x \in R$ be a real number defined on a specific interval $x \in [a, b]$. The opposite number $\tilde{x}$ is defined according to the following formula*

$$\tilde{x} = a + b - x \tag{4}$$

Definition 2. *Let $X_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$ be a point in D dimensional space, $x_{ij} \in [a_{ij}, b_{ij}]$, $j = 1, 2, \cdots, D$. The opposite point $\tilde{X}_i = (\tilde{x}_{i1}, \tilde{x}_{i2}, \cdots, \tilde{x}_{iD})$ is defined by*

$$\tilde{x}_{ij} = a_{ij} + b_{ij} - x_{ij} \tag{5}$$

Experiments show that, if there is no prior knowledge of optimization problem, the probability that the opposite solution can reach the global optimum is higher than that of the random solution [47]. Based on the OBL, quasi-opposition based learning [48] and quasi-reflective based learning [49] are proposed later. In this paper, we only consider OBL.

Taking TSP as an example, its solution is a sequence of numbers as the indices of cities. In addition, according to the opposite solutions for a continuous domain, it is challenging to construct opposite solutions for TSP due to the features of a discrete domain. Therefore, only a few scholars have made contributions towards this topic, and Ergeze is one of them. In [43], Ergeze addresses the definition of opposite paths according to the moving direction. For example, the initial path for n cities is given by

$$P = [1, 2, \cdots, n] \tag{6}$$

where the entries stand for the order of the cities that a salesman travels through. Then, the corresponding opposite path in a clockwise (CW) direction could be given by

$$P^{\mathrm{CW}} = \left[1, 1 + \frac{n}{2}, 2, 2 + \frac{n}{2}, \cdots, \frac{n}{2} - 1, n - 1, \frac{n}{2}, n\right] \tag{7}$$

where n is even.

In the case when n is odd, append an auxiliary city and make n even. In the end, find opposite solutions according to Equation (7) and then remove this city. Since different moving directions may lead to different opposite paths or solutions, moving in a counterclockwise (CCW) will result in different opposite solutions compared with P^{CW}.

When the number of cities is odd, one way of implementing CW opposition is to add an auxiliary city to the end of the path. After the opposite path is found, remove the auxiliary city.

3. Opposition-Based ACO

The method of construction opposite path based on OBL is given in this section. At the same time, in order to use the opposite path information, three kinds of frameworks of opposite-based ACO algorithms including ACO-Index, ACO-MaxIt, and ACO-Rand will also be proposed. In order to unify the content, the construction method of the opposite path will be combined with the specific algorithm. Details will be given in the following subsections.

3.1. ACO-Index

According to the definition given in Equation (7), the same route may lead to different opposite paths. Taking a TSP of six cities as an example, path $(1, 2, 3, 4, 5, 6)$ and path $(2, 3, 4, 5, 6, 1)$ are the same path; however, their opposite paths, $(1, 4, 2, 5, 3, 6)$ and $(2, 5, 3, 6, 4, 1)$, are different.

In addition, the initialization procedure of ACO is not random compared with DE, but more similar to the greedy algorithm, which selects a closer according to the rule of state transition. Therefore, opposite paths are always longer than the original ones generally and cannot be used pheromone updating. Aiming to solve the shortcomings, a novel ACO algorithm, namely ACO-Index, is proposed based on a modified strategy of opposite path construction.

Opposite path construction is mainly composed of two steps. The first step is the path sorting, and the second is the decision of opposite path. Suppose the number of cities n is even, then, during path sorting, put the path P back into a cycle and appoint a particular city A as the starting city with index 1. In addition, the rest of the cities will be given indices according to their position in this cycle. In this way, we could get the indices $P_{\text{ind}} = [1, 2, ..., n]$.

During the second step, indices of the opposite path $P_{\text{ind}}^{\text{CW}}$ should be found through Equation (7) and the opposite path P^{CW} can be found based on the indices $P_{\text{ind}}^{\text{CW}}$ appointed previously.

Moreover, when the number of cities n is odd, an auxiliary index should be added to the end of the indices P_{ind}, and we could get P_{aux}. According to Equation (7), we could get the opposite indices $P_{\text{aux}}^{\text{CW}}$, and its last index is the auxiliary index itself. Remove the latest index, and we can get $P_{\text{ind}}^{\text{CW}}$. In addition, then, decide the opposite path P^{CW} according to the opposite indices $P_{\text{ind}}^{\text{CW}}$.

In this way, opposite paths for different paths that share the same cycle route are the same. Pseudocode for opposite path construction is addressed in Algorithm 1.

Algorithm 1 Constructing the opposite path

Input: original path P
Put the path back into a circle
Appoint a specific city A with index 1
Appoint other cities in this circle with indices 2, 3, $\cdots$, and get the indices $P_{\text{ind}} = [1, 2, \cdots, n]$
if n is **even**
    Calculate the opposite indices $P_{\text{ind}}^{\text{CW}}$ according to Equation (7)
else
    Add an auxiliary index at the end of P_{ind} and get P_{aux}
    Calculate the opposite indices $P_{\text{aux}}^{\text{CW}}$ according to Equation (7)
    Delete the final index from $P_{\text{aux}}^{\text{CW}}$ and get $P_{\text{ind}}^{\text{CW}}$
endif
Calculate the P^{CW} based on $P_{\text{ind}}^{\text{CW}}$
Output: opposite path P^{CW}

Although some paths may be longer than the optimal path, they still contain useful information within themselves, which inspires us to apply them to reasonably modifying pheromone. For ACO algorithms, if the number of ants is m, the number of paths should also be m for pheromone updating. In the proposed ACO-Index, the top m_1 shortest original paths and the top m_2 shortest opposite paths will be chosen to form the $m = m_1 + m_2$ paths. Algorithm 2 presents the pseudocode for pheromone updating.

Algorithm 2 Updating pheromone

Input: original paths and opposite paths
Sort original paths and opposite paths by length
Select the top m_1 shortest paths and the top m_2 shortest opposite paths
Construct $m = m_1 + m_2$ new paths
Update pheromone according to Equation (2)
Output: Pheromone trail in each path

Algorithm 3 shows the pseudocode for the primary steps of ACO-Index for total iterations $N_{\max}$.

Algorithm 3 ACO-Index algorithm

Input: parameters: m, n, α, β, ρ, Q, m_1, m_2, $N_{\max}$
Initialize pheromone and heuristic information
for iteration index $N_c \leq N_{\max}$ **do**
  for $k = 1$ to m **do**
    Construct paths according to Equation (1)
    Construct opposite paths through Algorithm 1
  endfor
  Update pheromone according to Algorithm 2
endfor
Output: the optimal path

3.2. ACO-MaxIt

Although ACO-Index modifies ACO with a better path construction strategy, it inherits a similar opposite path generation method from [43]. In this section, a novel opposite path generation method, together with a novel pheromone updating rule, is proposed as an improved ACO algorithm, named ACO-MaxIt, which will be described in detail as follows.

The mirror point M is defined by

$$M = \left\lceil \frac{1+n}{2} \right\rceil \tag{8}$$

where $\lceil \cdot \rceil$ denotes the ceiling operator.

Considering the case when n is odd, the opposite city $\tilde{C}$ for the current city C could be defined as follows:

$$\tilde{C} = \begin{cases} C, & \text{if } C = M \\ C + M, & \text{if } C < M \\ C - M, & \text{if } C > M \end{cases} \tag{9}$$

Considering the case when n is even, the opposite city $\tilde{C}$ for the current city C could be defined as follows

$$\tilde{C} = \begin{cases} C, & \text{if } C = n/2 \text{ or } (n/2 + 1) \\ C + M, & \text{else if } C < M \\ C - M, & \text{else if } C > M \end{cases} \tag{10}$$

The pseudocode for opposite path construction is shown in Algorithm 4.

Algorithm 4 Constructing the opposite path based on the mirror point

Input: original path
Decide mirror point M, according to Equation (8)
for $C = 1$ to n **do**
  Calculate $\tilde{C}$ through Equation (9) or Equation (10) according to the parity of n
endfor
Output: opposite path

The pheromone update process consists of two stages. For the first stage, when $N_c \leq gN_{\max}$ and $0 < g < 1$, opposite paths will be decided through Algorithm 4. Meanwhile, the pheromone will be updated according to Algorithm 2. In the later stage, when $N_c > gN_{\max}$, no more opposite paths could be calculated, and pheromones will still be updated according to Equation (2).

The pseudocode of ACO-MaxIt is presented in Algorithm 5.

Algorithm 5 ACO-MaxIt algorithm

Input: parameters: m, n, α, β, ρ, Q, g, m_1, m_2, $N_{\max}$
Initialize pheromone and heuristic information
for iteration index $N_c \leq N_{\max}$ **do**
  for $k = 1$ to m **do**
    Construct paths according to Equation (1)
  endfor
  if $N_c \leq gN_{\max}$ **then**
    Construct opposite paths according to Algorithm 4
    Update pheromone according to Algorithm 2
  else
    Update pheromone according to Equation (2)
  endif
endfor
Output: the optimal solution

3.3. ACO-Rand

In the pheromone updating stage of ACO-Index or ACO-MaxIt, it is decided based on experiences of when to calculate the opposite paths. Therefore, in this section, another strategy to update pheromones is addressed, and the novel ACO algorithm is named ACO-Rand since whether or not to construct the opposite path is decided by two random variables.

The whole procedure of ACO-Rand is much like that of ACO-MaxIt; however, two random variables R_0 and R are introduced. R_0 is chosen randomly but fixed after generated, and R is randomly selected during each iteration. The pseudocode of ACO-Rand is given in Algorithm 6.

Algorithm 6 ACO-Rand algorithm

Input: parameters: m, n, α, β, ρ, Q, m_1, m_2, $N_{\max}$, R_0
Initialize pheromone and heuristic information
for iteration index $N_c \leq N_{\max}$ **do**
  for $k = 1$ to m **do**
    Construct paths according to Equation (1)
  endfor
  Generate a random variable R
  if $R < R_0$ **then**
    Construct opposite paths according to Algorithm 4
    Update pheromone according to Algorithm 2
  else
    Update pheromone according to Equation (2)
  endif
endfor
Output: the optimal solution

3.4. Time Complexity Analysis

The main steps of the three improved ant colony algorithms include initialization, solution construction and pheromone updating. The time complexity of initialization is $O(n^2 + m)$. The time complexity of constructing the solution is $O(mn^2)$. The time complexity of pheromone updating is $O(n^2)$. In addition, the time complexity of constructing and sorting the inverse solutions is $O(n^2)$. Therefore, the complexity of the final algorithm is $O(N_{max}mn^2)$. It is the same time complexity as the basic ant colony algorithm. Therefore, the improved algorithm does not increase significantly in time.

4. Experiments and Results

AS and PS-ACO are employed as ACO algorithms to verify the feasibility of three opposition-based ACO algorithms. The experiments were performed in the following hardware and software environments. CPU is Core i5@2.9 GB, and RAM is 16 GB. The operating system is Windows 10. TSP examples are exported from TSPLIB (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/).

4.1. Parameter Setting

In the following experiments, the parameters are setting as $m = 50$, $m_1 = 40$, $m_2 = 10$, $\alpha = 1$, $\beta = 2$, ρ=0.05, Q=1, $N_{\max} = 2000$, $g = 0.5$ for ACO-MaxIt, $R_0 = 0.6$ while $R \in (0, 1)$ for ACO-Rand. Twenty cycles of experiments are carried out for each example independently. Then, minimum solution $S_{\min}$, maximum solution $S_{\max}$, average solution S_{avg}, standard deviation S_{td}, and average runtime T_{avg} for different examples of 20 times are given in the tables, where minimum solution $S_{\min}$, maximum solution $S_{\max}$, and average solution S_{avg} are the percentage value deviation against the known optimal solution. The minimum value in each result is bolded in the tables.

4.2. Experimental Results Comparison Based on AS

First, we employ AS to three kinds of opposite based ACO, called AS-Index, AS-MaxIt, and AS-Rand, to verify the effectiveness of the improved algorithm. Twenty-six TSP examples are divided into three main categories, the small-scale, the medium-scale, and the large-scale according to the number of cities, respectively.

Small-scale city example sets are selected from TSPLIB, including eil51, st70, pr76, kroA100, eil101, bier127, pr136, pr152, u159, and rat195. The results are shown in Table 1.

From Table 1, the proposed AS-Index, AS-MaxIt, and AS-Rand show superior performances over AS for the examples, st70, kroA100, eil101, bier127, pr136, and u159. For other examples, the proposed algorithms outperform AS in general, except eil51. Meanwhile, stability by standard deviation is the not the primary concern when evaluating an algorithm. Compared among three proposed algorithms, AS-MaxIt illustrates superior performances for most cases.

To show more details in the process of evolutionary, curves for different stages are given in Figure 1 based on the case of kroA100.

According to Figure 1, AS shows faster convergence speed than the other three proposed methods in early iterations, while AS-Index, AS-MaxIt, and AS-Rand all surpass AS in average path length in later iterations. Meanwhile, AS-MaxIt performs best among all these algorithms, which also verifies the results in Table 1.

In the early stage, opposite path information introduced by OBL has negative impact on the convergence speed for all three proposed algorithms; however, it can provide extra information which guarantees the boost in accuracy for the later stage. The results lie in the fact that introducing extra information of opposite paths help to increase the diversity of the population, which balances the exploration and exploitation of solution space.

Medium-scale city example sets are selected from TSPLIB, including kroA200, ts225, tsp225, pr226, pr299, lin318, fl417, pr439, pcb442, and d493. The results are shown in Table 2.

From Table 2, it can be found that the proposed algorithms outperform AS in all the cases except ts225. Among all three algorithms, AS-Index and AS-MaxIt perform similarly, but better than AS-Rand generally. From these results, it can be seen that, with the help of extra information from opposite paths, three proposed methods all improve the original AS in solution accuracy.

Table 1. Results comparison for small-scale example sets.

Instance	Algorithm	$S_{\min}$(%)	$S_{\max}$(%)	S_{avg}(%)	S_{td}	T_{avg}
eil51	AS	**2.58**	6.1	**3.56**	4.94	65.21
	AS-Index	2.81	7.51	4.27	5.26	64.73
	AS-MaxIt	**2.58**	7.04	4.25	6.66	**63.1**
	AS-Rand	2.82	**5.63**	3.86	**4.67**	63.76
st70	AS	5.03	7.41	6.22	5.16	90.38
	AS-Index	**3.85**	7.56	6.03	7.05	104.02
	AS-MaxIt	4.59	**6.81**	**5.83**	**4.63**	89.87
	AS-Rand	4.59	8.15	6.25	5.45	**85.29**
pr76	AS	6.03	9.11	7.49	**897.25**	101.6
	AS-Index	6.22	9.83	7.81	926.53	118.32
	AS-MaxIt	5.04	**8.46**	**6.66**	1151.2	95.9
	AS-Rand	**4.71**	8.78	7.18	1336	**95.72**
kroA100	AS	4.86	6.78	5.32	94.22	146.29
	AS-Index	**4.29**	6.36	4.93	111.14	163.71
	AS-MaxIt	4.35	**5.66**	**4.79**	80.21	**123.84**
	AS-Rand	4.53	5.8	5.11	**56.40**	124.99
eil101	AS	8.11	12.1	9.99	**5.78**	140.25
	AS-Index	**6.2**	**10.81**	**8.55**	9.55	140.2
	AS-MaxIt	7.15	11.8	9.3	6.86	**128.16**
	AS-Rand	7.15	12.4	9.96	8.23	145.56
bier127	AS	4.75	7.15	6.05	**828.19**	195.33
	AS-Index	4.07	**6.75**	5.32	870.88	176.32
	AS-MaxIt	**3.34**	6.82	**5.03**	904.32	**173.58**
	AS-Rand	3.52	6.8	5.05	961.25	175.14
pr136	AS	9.83	13.5	11.82	924.44	**189.68**
	AS-Index	9.64	12.47	11.47	**832.93**	209.51
	AS-MaxIt	**8.13**	**12.34**	**10.73**	976.21	190.95
	AS-Rand	10.68	12.62	10.95	890.29	191.23
pr152	AS	4.3	7.34	5.79	552.91	214.30
	AS-Index	3.56	8.03	6.25	804.54	238.72
	AS-MaxIt	**3.49**	**6.83**	5.33	739.76	**1982**
	AS-Rand	4.16	6.95	**5.14**	**537.49**	220.52
u159	AS	7.67	10.3	6.86	**348.22**	219.92
	AS-Index	6.31	8.97	7.44	349.95	223.39
	AS-MaxIt	**3.85**	**8.32**	**6.28**	479.93	**206.23**
	AS-Rand	4.88	8.55	7.25	412.44	229.64
rat195	AS	3.92	9.38	7.43	35.28	286.38
	AS-Index	**3.49**	8.52	6.47	37.02	283.31
	AS-MaxIt	3.83	7.58	5.59	37.69	**278.63**
	AS-Rand	4.00	**6.54**	**5.31**	**17.01**	280.39

Taking fl417 as the example, evolutionary curves in detail for different iteration stages are given in Figure 2, accordingly. According to Figure 2, AS also converges faster than the other three proposed methods in early iterations—for example before 1000 iterations. In addition, in later iterations, the other three proposed methods all exceed AS in average path length. This further validates the conclusions obtained from the analysis of Figure 1.

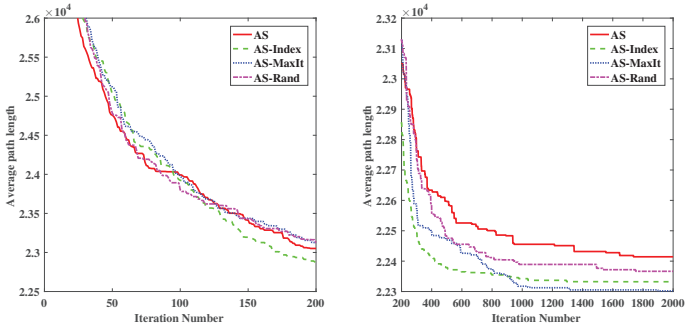


Figure 1. Evolutionary curves for different iteration periods based on kroA100.

Table 2. Results comparison for medium-scale example sets.

Instance	Algorithm	$S_{\min}$(%)	$S_{\max}$(%)	S_{avg}(%)	S_{td}	T_{avg}
kroA200	AS	10.33	16.54	12.36	454.71	330.43
	AS-Index	10.17	13.06	11.12	**243.66**	324.495
	AS-MaxIt	8.09	13.48	11.27	367.58	**270.07**
	AS-Rand	**6.69**	**13.01**	**10.66**	485.83	292.98
ts225	AS	3.53	**4.27**	**3.89**	**275.55**	345.62
	AS-Index	**3.13**	4.74	3.91	488.06	329.03
	AS-MaxIt	3.35	4.94	4.01	560.16	329.35
	AS-Rand	3.63	5.52	4.23	607.16	**325.54**
tsp225	AS	9.4	**12.03**	10.37	**30.67**	346.81
	AS-Index	8.02	12.16	**9.9**	38.02	345.35
	AS-MaxIt	**7.51**	12.87	10.91	46.8	**319.75**
	AS-Rand	9.5	12.39	10.88	36.13	355.0
pr226	AS	5.5	7.94	6.76	632.0	327.3
	AS-Index	4.96	7.28	6.47	**458.47**	348.97
	AS-MaxIt	**4.29**	**7.24**	6.34	631.53	335.54
	AS-Rand	4.39	7.5	**5.92**	578.15	**325.31**
pr299	AS	13.31	19.48	17.03	732.30	512.6
	AS-Index	**9.43**	**17.41**	**14.95**	1193.8	495.6
	AS-MaxIt	15.53	18.24	16.91	**419.72**	**487.8**
	AS-Rand	11.35	18.73	16.41	836.03	500.35
lin318	AS	12.57	17.15	15.33	517.72	**508.6**
	AS-Index	**11.8**	17.16	**14.97**	544.18	554.6
	AS-MaxIt	13.93	**16.89**	15.59	**376.06**	542.2
	AS-Rand	14.28	17.74	16.22	415.81	542.55
fl417	AS	8.16	12.55	10.61	132.22	811.45
	AS-Index	**7.79**	12.57	**9.91**	141.28	804.6
	AS-MaxIt	8.35	**11.55**	10.3	**108.13**	**773**
	AS-Rand	8.67	13.24	10.39	146.72	795.35
pr439	AS	9.5	13.74	11.56	1410.4	881.25
	AS-Index	**8.38**	**11.75**	**10.22**	**1010.9**	845.75
	AS-MaxIt	9.34	15.96	11.8	1555.1	**841.9**
	AS-Rand	11.73	15.91	13.76	1205.6	859.8
pcb442	AS	13.57	18.87	16.84	**610.98**	933.5
	AS-Index	**11.62**	**16.37**	**14.22**	640.69	930.7
	AS-MaxIt	13.68	19.27	16.66	656.89	899.35
	AS-Rand	14.54	18.85	16.68	644.38	**888.45**
d493	AS	13.4	19.18	16.26	459.03	1032.25
	AS-Index	**12.23**	**16.31**	**14.71**	412.28	1043.25
	AS-MaxIt	14.08	17.01	15.7	**280.32**	**969.5**
	AS-Rand	13.46	19.21	16.4	449.97	1063.8

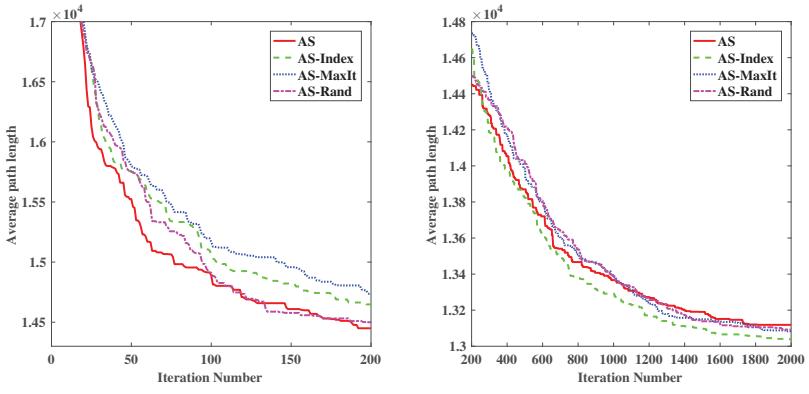


Figure 2. Evolutionary curves for different iteration periods based on fl417.

Large-scale city example sets are selected from TSPLIB, including att532, rat575, d657, u724, vm1084, and rl1304. The results are shown in Table 3.

From Table 3, it can be discovered that the AS-Index shows the obvious superior performance over all the other algorithms, which reveals a fact that the advantages of AS-Index appears as the scale of the example increases based on these results.

Taking vm1084 as the example, evolutionary curves in detail for different iteration stages are given in Figure 3, accordingly. According to Figure 3, AS still shows faster convergence speed than the other three proposed methods in early iterations, but AS-Index outperforms all the others in the end.

Based on all the tables and figures, it can be found that, in most scenarios, at least one of AS-Index, AS-MaxIt, and AS-Rand outperforms AS in average path length. For small-scale examples, AS-MaxIt shows better performance, while, for medium-scale cases, AS-Index and AS-MaxIt perform similarly better than the others. For large-scale city sets, AS-Index is the best algorithm, while AS-Rand ranks in the middle for most cases regarding figures, and it illustrates its stability to some extent. Therefore, it can be drawn that the strategy to introduce OBL into AS provides more information, namely better exploration capability, which explains the superiority of these proposed methods over the original AS. By comparing the results of the running time from Tables 1–3, we can also find that the running time of the three improved algorithms is not significantly increased compared with AS. It also validates our previous discussion on time complexity.

4.3. Experimental Results Comparison Based on PS-ACO

To further verify the effectiveness of the proposed algorithm, we employed another PS-ACO to three kinds of opposite based ACO, PS-ACO-Index, PS-ACO-MaxIt, and PS-ACO-Rand to verify the effectiveness of the improved algorithm. The number of ants is 50, and the other parameters are the same as in [26]. Twelve sets of TSP examples are eil51, st70, kroA100, pr136, u159, rat195, tsp225, pr299, lin318, fl417, att532, and d657. The results are given in Table 4.

From Table 4, the proposed PS-ACO-Index, PS-ACO-MaxIt, and PS-ACO-Rand show superior performances over PS-ACO for the examples, eil51, st70, rat195, tsp225, and pr299. For other examples, the proposed algorithms outperform PS-ACO in general, except lin318 and fl417. Compared among three proposed algorithms, PS-ACO-Rand illustrates superior performances for most cases. By comparing the results of the running time, we can also find that the running time of the three improved algorithms is not significantly increased compared with PS-ACO.

Table 3. Results comparison for large-scale example sets.

Instance	Algorithm	$S_{\min}(\%)$	$S_{\max}(\%)$	$S_{avg}(\%)$	S_{td}	T_{avg}
att532	AS	13.79	20.21	17.3	1415.4	1407.05
	AS-Index	**13.37**	19.49	**15.64**	1203.4	1436.4
	AS-MaxIt	14.33	18	16.15	**766.9**	**1384.65**
	AS-Rand	14.63	**17.98**	16.34	803.93	1460.35
rat575	AS	18.69	22.52	20.5.3	75.22	1651.5
	AS-Index	**16.31**	**20.71**	**18.94**	**61.3**	1661.7
	AS-MaxIt	20.24	24.32	22.23	80.88	1602.55
	AS-Rand	22.88	26.93	25.36	86.12	**1598**
d657	AS	17.6	23.88	21.97	823.36	2685.45
	AS-Index	**16.32**	**21.5**	**19.7**	636.0	2673.65
	AS-MaxIt	19.4	24.09	21.69	**526.44**	**2114.15**
	AS-Rand	18.21	23.82	21.24	615.17	2138.1
u724	AS	20.9	26.43	24.19	625.6	2630.95
	AS-Index	**15.53**	**23.45**	**20.46**	871.17	2624.7
	AS-MaxIt	21.6	26.72	24.05	**469.3**	2651.65
	AS-Rand	19.74	27.13	24.3	825.26	**2609.7**
vm1084	AS	22.65	27.78	25.76	3490.5	6722
	AS-Index	**17.69**	**24.73**	**21.1**	4700	6726.5
	AS-MaxIt	19.91	26.59	23.29	4476.6	**6594**
	AS-Rand	20.72	25.71	23.04	**3111**	6695.5
rl1304	AS	19.51	24.37	21.76	3633.6	7436.5
	AS-Index	**16.65**	**21.63**	**18.95**	**3632.3**	**7383**
	AS-MaxIt	18.45	24.19	20.63	3709.8	8767.5
	AS-Rand	17.09	22.82	20.12	4359.2	8652.5

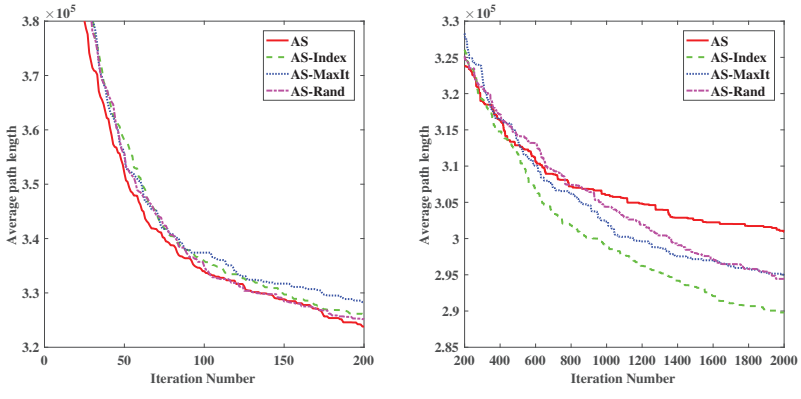


Figure 3. Evolutionary curves for different iteration periods based on vm1084.

Table 4. Comparisons of PS-ACO, PS-ACO-Index, PS-ACO-MaxIt, and PS-ACO-Rand.

Instance	PS-ACO					PS-ACO-Index					PS-ACO-MaxIt					PS-ACO-Rand				
	$S_{\min}(\%)$	$S_{\max}(\%)$	$S_{avg}(\%)$	S_{td}	T_{avg}	$S_{\min}(\%)$	$S_{\max}(\%)$	$S_{avg}(\%)$	S_{td}	T_{avg}	$S_{\min}(\%)$	$S_{\max}(\%)$	$S_{avg}(\%)$	S_{td}	T_{avg}	$S_{\min}(\%)$	$S_{\max}(\%)$	$S_{avg}(\%)$	S_{td}	T_{avg}
eil51	0	0.7	0.52	0.89	77.4	0	0.7	0.42	1.15	81.59	0	0.7	0.46	1.15	78.06	0	0.7	0.46	0.94	78.58
st70	0.15	2.67	1.07	5.84	99.35	0	2.67	0.88	4.67	121.44	0.15	2.96	0.93	5.61	99.31	0.15	692	2.52	3.79	99.02
kroA100	0.06	1.16	0.54	58.85	163.6	0.12	2.2	0.72	107.74	176.34	0.06	1	0.55	59.94	151.06	0	1.11	0.54	79.47	146.32
pr136	7.64	12.2	10.08	1200	206.41	7.98	13.12	10.19	1452.7	230.68	8.03	11.97	9.96	1116.4	203.95	7.48	12.51	9.81	1274.4	204.24
u159	0	0.88	0.12	112.74	886.25	0	0.88	0.17	117.7	875.5	0	0.22	0.02	28.93	860.8	0	0.75	0.09	98.07	868.45
rat195	0.43	1.33	0.7	5.86	882.25	0.43	1.21	0.6	4.24	875.65	0.43	1.33	0.6	6.45	907.85	0.43	0.99	0.57	3.88	906.4
tsp225	3.7	5.98	4.93	23.78	1087.35	3.5	5.36	4.21	21.14	1140.75	3.73	5.87	4.66	25.18	1158.85	3.52	5.57	4.76	20.51	1150.8
pr299	9.78	14.08	11.68	605.79	1927.8	9.03	12.78	10.63	450.92	1919.15	7.8	14.74	11.14	885.13	1882.4	9.17	14.49	11.6	792.39	1881.45
lin318	10.29	15.59	12.53	523.89	571.9	11.42	15.14	13.16	417.59	651.4	9.66	15.64	12.85	600.33	608.75	10.64	15.71	13.52	498.17	607.8
fl417	10.51	14.56	12.43	155.86	782.6	11.1	17.45	14.7	226.592	778.95	11.45	17.77	14.17	216.73	714.3	10.35	17.56	14.45	232.11	724.6
att532	19.9	25.06	22.82	1106.8	1518.45	19.58	25.26	22.42	1372.8	1529.25	20.02	25.3	22.3	1113.1	1342.4	20.59	26.17	23.25	1114	1363.95
d657	22.67	27.06	24.66	534.7	2247.4	21.04	27.91	24.57	780.93	2202.4	22.08	25.74	24.32	547.2	2002.5	23.17	28.16	25.31	697.21	2175.95

5. Conclusions

The performances of swarm optimization algorithms based on OBL present advantages when handling problems of continuous optimization. However, there are only a few approaches proposed to solve problems of discrete optimization. The difficulty in opposite solution construction is considered as one top reason. To solve this problem, two different strategies, direction and indirection, of constructing opposite paths are presented individually in this paper. For indirection strategy, other than using the order of cities from the current solution directly, it studies the positions, noted as indices, of the cities rearranged in a circle, and then calculates the opposite indices. While for direction strategy, opposite operations are carried out directly to the cities in each path.

To use the information of the opposite path, three different frameworks of opposite-based ACO, called ACO-Index, ACO-MaxIt, and ACO-Rand, are also proposed. All ants need to get the increment of pheromone in three improved frameworks. Among three proposed algorithms, ACO-Index employs the strategy of indirection to construct the opposite path and introduces it to pheromone updating. ACO-MaxIt also employs direction strategy to obtain opposite path but only adopts it in the early updating period. Similar to ACO-MaxIt in opposite path construction, ACO-Rand employs this opposite path throughout the stage of pheromone updating. In order to verify the effectiveness of the improvement strategy, AS and PS-ACO are used in three frameworks, respectively. Experiments demonstrate that all three methods, As-Index, As-MaxIt, and AS-Rand, outperform original AS in the cases of small-scale and medium-scale cities while AS-Index performs best when facing large-scale cities. The three improved PS-ACO also showed good performance.

Constructing the opposite path mentioned in this paper is only suitable for symmetric TSP. This is mainly because the path (solution) of the problem is an arrangement without considering the direction. However, if it is replaced by the asymmetric TSP, this method needs to be modified. In addition, if it is replaced by a more general combinatorial optimization problem, it is necessary to restudy how to construct the opposite solution according to the characteristics of the problem. Therefore, our current method of constructing opposite solution is not universal. This is one of the limitations of this study. At the same time, the improved algorithm requires all ants to participate in pheromone updating in order to use the information of opposite path. However, now many algorithms use the best ant to update pheromone, so the method in this paper will have some limitations when it is extended to more ant colony algorithm. However, we also find that it is effective to apply reverse learning to combinatorial optimization problems. Therefore, we will carry out our future research work from two aspects. On the one hand, we plan to continue to study the construction method of more general opposite solution for combinatorial optimization problems, so as to improve its generality. In addition, it will be applied to practical problems such as path optimization to further expand the scope of application. Meanwhile, applying OBL to more widely used algorithms is also one interesting and promising topic. Therefore, on the other hand, we plan to study more effective use of the reverse solution and extend it to the more wildly used ACO, such as MMAS and ACS, and even some other optimization algorithms such as PSO and ABC, to solve more combinatorial optimization problems more effectively.

Author Contributions: Conceptualization, Z.Z.; methodology, Z.Z.and Z.X.; software, Z.X. and X.L.; formal analysis, Z.Z.and Z.X.; resources, Z.Z.; writing—original draft preparation, Z.X.; writing—review and editing, Z.Z.and S.L.; supervision, Z.Z.and Y.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (Grant No. 61703256, 61801197), Jiangsu Natural Science Foundation (Grant No. BK20181004), Natural Science Basic Research Plan In Shaanxi Province of China (Program No. 2017JQ6070), and the Fundamental Research Funds for the Central Universities (Grant No. GK201603014, GK201803020).

Acknowledgments: The authors are grateful to the anonymous reviewers and the editor for the constructive comments and valuable suggestions.

Conflicts of Interest: The authors declare no conflict of interest.The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Karaboga, D.; Akay, B. A survey: Algorithms simulating bee swarm intelligence. *Artif. Intell. Rev.* **2009**, *31*, 61–85.
2. Dorigo,M.; Maniezzo,V.; Colorni,A. The ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. Part B* **1996**, *26*, 29–41. [CrossRef]
3. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
4. Karaboga, D.; Akay, B. A comparative study of artificial bee colony algorithm. *Appl. Math. Comput.* **2009**, *214*, 108–132. [CrossRef]
5. Yang, X.S. Firefly algorithm, stochastic test functions and design optimisation. *Int. J. Bio-Inspired Comput.* **2010**, *2*, 78–84. [CrossRef]
6. Gandomi, A.H.; Yang, X.S.; Alavi, A.H. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Eng. Comput.* **2013**, *29*, 17–35. [CrossRef]
7. Wang, G.G.; Guo, L.G.; omi, A.H.; Hao, G.S.; Wang, H. Chaotic krill herd algorithm. *Inf. Sci.* **2014**, *274*, 17–34. [CrossRef]
8. Wang, G.G.; Deb, S.; Cui, Z. Monarch butterfly optimization. *Neural Comput. Appl.* **2019**, *31*, 1995–2014. [CrossRef]
9. Wang, G.G. Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Memet. Comput.* **2018**, *10*, 151–164.
10. Mollajafari, M.; Shahhoseini, H.S. An efficient ACO-based algorithm for scheduling tasks onto dynamically reconfigurable hardware using TSP-likened construction graph. *Appl. Intell.* **2016**, *45*, 695–712. [CrossRef]
11. Elloumi, W.; El Abed, H.; Abraham, A.; Alimi, A.M. A comparative study of the improvement of performance using a PSO modified by ACO applied to TSP. *Appl. Soft Comput.* **2014**, *25*, 234–241. [CrossRef]
12. Zhang, Z.Z.; Hu, F.N.; Zhang, N.; Ant colony algorithm for satellite control resource scheduling problem. *Appl. Intell.* **2018**, *48*, 3295–3305. [CrossRef]
13. Rahim, S.; Javaid, N.; Ahmad, A.; Khan, S.A.; Khan, Z.A.; Alrajeh, N.; Qasim, U. Exploiting heuristic algorithms to efficiently utilize energy management controllers with renewable energy sources. *Energy Build.* **2016**, *129*, 452–470. [CrossRef]
14. Bhattacharjee, K.K.; Sarmah, S.P. Modified swarm intelligence based techniques for the knapsack problem. *Appl. Intell.* **2017**, *46*, 158–179. [CrossRef]
15. Huang, S.H.; Huang, Y.H.; Blazquez, C.A.; Paredes-Belmar, G. Application of the ant colony optimization in the resolution of the bridge inspection routing problem. *Appl. Soft Comput.* **2018**, *65*, 443–461. [CrossRef]
16. Lee, C.Y.; Lee, Z.J.; Lin, S.W.; Ying, K.C. An enhanced ant colony optimization (EACO) applied to capacitated vehicle routing problem. *Appl. Intell.* **2010**, *32*, 88–95. [CrossRef]
17. Kumar, A.; Thakur, M.; Mittal, G. A new ants interaction scheme for continuous optimization problems. *Int. J. Syst. Assur. Eng. Manag.* **2018**, *9*, 784–801. [CrossRef]
18. Yang, Q.; Chen, W.N.; Yu, Z.; Gu, T.; Li, Y.; Zhang, H.; Zhang, J. Adaptive multimodal continuous ant colony optimization. *IEEE Trans. Evol. Comput.* **2017**, *21*, 191–205. [CrossRef]
19. Liao, T.J.; Stützle, T.; Oca, M.A.M.; Dorigo, M. A unified ant colony optimization algorithm for continuous optimization. *Eur. J. Oper. Res.* **2014**, *234*, 597–609. [CrossRef]
20. Dorigo, M.; Blum, C. Ant colony optimization theory: A survey. *Theor. Comput. Sci.* **2005**, *344*, 243–278. [CrossRef]
21. Dorigo, M.; Gambardella, L.M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evol. Comput.* **1997**, *1*, 53–66. [CrossRef]
22. Stützle, T.; Hoos, H.H. Max-min ant system. *Future Gener. Comput. Syst.* **2000**, *16*, 889–914.
23. Luo, Q.; Wang, H.; Zheng, Y.; He, J. Research on path planning of mobile robot based on improved ant colony algorithm. *Future Gener. Comput. Syst.* **2020**, *32*, 1555–1566. [CrossRef]
24. Huang, M.; Ding, P. An improved ant colony algorithm and its application in vehicle routing problem. *Future Gener. Comput. Syst.* **2013**, *2013*, 1–9.
25. Deng, Y.; Zhu, W.; Li, H.; Zheng, Y.H. Multi-type ant system algorithm for the time dependent vehicle routing problem with time windows. *J. Syst. Eng. Electron.* **2018**, *29*, 625–638.
26. Shuang, B.; Chen, J.; Li, Z. Study on hybrid PS-ACO algorithm. *Appl. Intell.* **2011**, *34*, 64–73. [CrossRef]

27. Ke, L.J.; Zhang, Q.F.; Battiti, R. MOEA/D-ACO: A multiobjective evolutionary algorithm using decomposition and ant colony. *IEEE Trans. Cybern.* **2013**, *43*, 1845–1859.
28. Akpınar, S.; Bayhan, G.M.; Baykasoglu, A. Hybridizing ant colony optimization via genetic algorithm for mixed-model assembly line balancing problem with sequence dependent setup times between tasks. *Appl. Soft Comput.* **2013**, *13*, 574–589.
29. Ting, T.O.; Yang, X.S.; Cheng, S.; Huang, K. Hybrid metaheuristic algorithms: past, present, and future. In *Recent Advances in Swarm Intelligence and Evolutionary Computation*; Yang, X.S., Ed.; Springer International Publishing: Cham, Switzerland, 2015; pp. 71–83.
30. Rosa, G.H.D.; Papa, J.P.; Yang, X.S. Handling dropout probability estimation in convolution neural networks using meta-heuristics. *Soft Comput.* **2018**, *22*, 6147–6156. [CrossRef]
31. Bacanin, N.; Bezdan, T.; Tuba, E.; Strumberger, I.; Tuba, M. Monarch butterfly optimization based convolutional neural network design. *Mathematics* **2020**, *8*, 936. [CrossRef]
32. Wang, G.G.; Tan, Y. Improving metaheuristic algorithms with information feedback models. *IEEE Trans. Cybern.* **2019**, *49*, 542–555. [CrossRef]
33. Gao, D.; Wang, G.G.; Pedrycz, W. Solving fuzzy job-shop scheduling problem using DE algorithm improved by a selection mechanism. *IEEE Trans. Fuzzy Syst.* **2020**. [CrossRef]
34. Li, W.; Wang, G.G.; Alavi, A.H. Learning-based elephant herding optimization algorithm for solving numerical optimization problems. *Knowl. Based Syst.* **2020**, *195*, 105675. [CrossRef]
35. Mahdavi, S.; Rahnamayan, S.; Deb, K. Opposition based learning: A literature review. *Swarm Evol. Comput.* **2017**, *39*, 1–23. [CrossRef]
36. Wang, B. A novel artificial bee colony algorithm based on modified search strategy and generalized opposition-based learning. *J. Intell. Fuzzy Syst.* **2015**, *28*, 1023–1037. [CrossRef]
37. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M.A. Opposition-based differential evolution. *IEEE Trans. Evol. Comput.* **2008**, *12*, 64–79. [CrossRef]
38. Chen, J.; Cui, G.; Duan, H. Multipopulation differential evolution algorithm based on the opposition-based learning for heat exchanger network synthesis. *Numer. Heat Transf. Part A Appl.* **2017**, *72*, 126–140. [CrossRef]
39. Park, S.Y.; Lee, J.J. Stochastic opposition-based learning using a beta distribution in differential evolution. *IEEE Trans. Cybern.* **2016**, *46*, 2184–2194. [CrossRef] [PubMed]
40. Dong, W.; Kang, L.; Zhang, W. Opposition-based particle swarm optimization with adaptive mutation strategy. *Soft Comput.* **2017**, *21*, 5081–5090. [CrossRef]
41. Kang, Q.; Xiong, C.; Zhou, M.; Meng, L. Opposition-based hybrid strategy for particle swarm optimization in noisy environments. *IEEE Access* **2018**, *6*, 21888–21900. [CrossRef]
42. Malisia, A.R.; Tizhoosh, H.R. Applying opposition-based ideas to the ant colony system. In Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS), Honolulu, HI, USA, 1–5 April 2007; pp. 182–189.
43. Ergezer, M.; Simon, D. Oppositional biogeography-based optimization for combinatorial problems. In Proceedings of the 2011 IEEE Congress of Evolutionary Computation (CEC), New Orleans, LA, USA, 5–8 June 2011; pp. 1496–1503.
44. Srivastava, G.;Singh, A. Boosting an evolution strategy with a preprocessing step: Application to group scheduling problem in directional sensor networks. *Appl. Intell.* **2018**, *48*, 4760-4774. [CrossRef]
45. Venkatesh, P.; Alok, S. A swarm intelligence approach for the colored traveling salesman problem. *Appl. Intell.* **2018**, *48*, 4412–4428.
46. Sarkhel, R.; Das, N.; Saha, A.K.; Nasipuri, M. An improved harmony search algorithm embedded with a novel piecewise opposition based learning algorithm. *Eng. Appl. Artif. Intell.* **2018**, *67*, 317–330. [CrossRef]
47. Wang, H.; Wu, Z.; Rahnamayan, S. Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems. *Soft Comput.* **2011**, *15*, 2127–2140. [CrossRef]
48. Guha, D.; Roy, P.K.; Banerjee, S. Load frequency control of large scale power system using quasi-oppositional grey wolf optimization algorithm. *Eng. Sci. Technol. Int. J.* **2016**, *19*, 1693–1713. [CrossRef]
49. Ewees, A.A.; Elaziz, M.A.; Houssein, E.H. Improved grasshopper optimization algorithm using opposition-based learning. *Expert Syst. Appl.* **2018**, *112*, 156–172. [CrossRef]

mathematics

MDPI

Article

Binary Whale Optimization Algorithm for Dimensionality Reduction

Abdelazim G. Hussien [1], Diego Oliva [2,3], Essam H. Houssein [4], Angel A. Juan [2] and Xu Yu [5,*]

[1] Faculty of Science, Fayoum University, Faiyum 63514, Egypt; aga08@fayoum.edu.eg
[2] IN3-Computer Science Department, Universitat Oberta de Catalunya, 08018 Barcelona, Spain; diego.oliva@cucei.udg.mx (D.O.); ajuanp@uoc.edu (A.A.J.)
[3] Departamento de Ciencias Computacionales, Universidad de Guadalajara, CUCEI, Guadalajara 44430, Mexico
[4] Faculty of Computers and Information, Minia University, Minia 61519, Egypt; essam.halim@mu.edu.eg
[5] School of Information Science and Technology, Qingdao University of Science and Technology, Qingdao 266061, China
* Correspondence: yuxu0532@qust.edu.cn

Received: 20 September 2020; Accepted: 12 October 2020; Published: 17 October 2020

Abstract: Feature selection (FS) was regarded as a global combinatorial optimization problem. FS is used to simplify and enhance the quality of high-dimensional datasets by selecting prominent features and removing irrelevant and redundant data to provide good classification results. FS aims to reduce the dimensionality and improve the classification accuracy that is generally utilized with great importance in different fields such as pattern classification, data analysis, and data mining applications. The main problem is to find the best subset that contains the representative information of all the data. In order to overcome this problem, two binary variants of the whale optimization algorithm (WOA) are proposed, called bWOA-S and bWOA-V. They are used to decrease the complexity and increase the performance of a system by selecting significant features for classification purposes. The first bWOA-S version uses the Sigmoid transfer function to convert WOA values to binary ones, whereas the second bWOA-V version uses a hyperbolic tangent transfer function. Furthermore, the two binary variants introduced here were compared with three famous and well-known optimization algorithms in this domain, such as Particle Swarm Optimizer (PSO), three variants of binary ant lion (bALO1, bALO2, and bALO3), binary Dragonfly Algorithm (bDA) as well as the original WOA, over 24 benchmark datasets from the UCI repository. Eventually, a non-parametric test called Wilcoxon's rank-sum was carried out at 5% significance to prove the powerfulness and effectiveness of the two proposed algorithms when compared with other algorithms statistically. The qualitative and quantitative results showed that the two introduced variants in the FS domain are able to minimize the selected feature number as well as maximize the accuracy of the classification within an appropriate time.

Keywords: whale optimization algorithm; WOA; Binary whale optimization algorithm; bWOA-S; bWOA-V; Feature selection; Classification; Dimensionality reduction

1. Introduction

The datasets from real-world applications such industry or medicine are high-dimensional and contain irrelevant or redundant features. These kind of datasets then have useless information that affects the performance of machine learning algorithms; in such cases, the learning process is affected. Feature selection (FS) is a powerful rattling technique used to select the most significant subset of features, overcoming the high-dimensionality reduction problem [1], identifying the relevant features and removing redundant ones [2]. Moreover, using the subset of features, any machine

learning algorithm can be applied for classification. Therefore, several studies have taken into consideration that the FS problem is an optimization problem, hence the fitness function for the optimization algorithm has been changed to classifier's accuracy, which may be maximized by the selected features [3]. Moreover, FS has been applied successfully to solve many classification problems in different domains, such as data mining [4,5], pattern recognition [6], information retrieval [7], information feedback [8], drug design [9,10], job-shop scheduling problem [11], maximizing lifetime of wireless sensor networks [12,13], and the others where FS can be utilized [14].

There are three main classes of FS methods: (1) The wrapper, (2) filter and (3) hybrid methods [15]. The wrapper approaches generally incorporate classification algorithms to search for and select the relevant features [16]. Filter methods calculate the relevant features without prior data classification [17]. In the hybrid techniques, the compatible strengths of the wrapper and filter methods are combined. Generally speaking, the wrapper methods outperform filter methods in terms of classification accuracy, and hence the wrapper approaches are used in this paper.

In fact, a high accuracy classification does not depend on a large selected features number for many classification problems. In this context, the classification problems can be categorized into two groups: (1) binary classification and (2) multi-class classification. In this paper, we deal with the binary classification problem. There are numerous methods that are applied for binary classification problems, such as discriminant analysis [18], decision trees (DT) [19], the K-nearest neighbor (K-NN) [20], artificial neural networks (ANN) [21], and support vector machines (SVMs) [22].

On the other hand, the traditional optimization methods suffer from some limitations in solving the FS problems [23,24], and hence nature-inspired meta-heuristic algorithms [25] such as the whale optimization algorithm (WOA) [26], moth–flame optimisation [27], Ant Lion Optimization [28], Crow Search Algorithm [29], Lightning Search Algorithm [30], Henry gas solubility optimization [31] and Lévy flight distribution [32] are widely used in the scientific community for solving complex optimization problems and several real-world applications [33–35]. Optimization is defined as a process of searching the optimal solutions to a specific problem. In order to address issues such as FS, several nature-inspired algorithms have been applied; some of these algorithms are hybridized with each other or used alone, others created new variants like binary methods to solve this problem. A survey on evolutionary computation [36] approaches for FS is presented in [37]. Several separate and hybrid algorithms have been proposed for FS, such as hybrid ant colony optimization algorithm [38], forest optimization algorithm [39], firefly optimization algorithm [40], hybrid whale optimization algorithm with simulated annealing [41], particle swarm optimization [42], sine cosine optimization algorithm [43], monarch butterfly optimization [44], and moth search algorithm [45].

In addition to the aforementioned studies to find solutions for the FS problem, other search strategies called the binary optimization algorithms have been implemented. Some examples are the binary flower pollination algorithm (BFPA) in [46], binary bat algorithm (BBA) in [47], binary cuckoo search algorithm (BCSA) in [48]; all of them evaluate the accuracy of the classifier as an objective function. He et al. have presented a binary differential evolution algorithm (BDEA) [49] to select the relevant subset to train a SVM with radial basis function (RBF). Moreover, Emary et al., have proposed the binary ant lion and the binary grey wolf optimization [50,51], respectively. Rashedi et al. have introduced an improved binary gravitational search algorithm version called (BGSA) [52]. In addition, a salps algorithm is used for feature selection of the chemical compound activities [53]. A binary version of particle swarm optimization (BPSO) is proposed [54]. A binary whale optimization algorithm for feature selection [55–57] has also been introduced. As the NO Free Lunch (NFL) theorm states, there is no algorithm that is able to solve all optimization problems. Hence, if an algorithm shows a superior performance on a class of problem, it cannot show the same performance on other classes. This is the motivation of our presented study, in which we propose two novel binary variants of the whale optimization algorithm (WOA) called bWOA-S and bWOA-V. In this regard, the WOA is a nature-inspired population-based metaheuristics optimization algorithm, which simulates the humpback whales' social behavior [26]. The original WOA was modified in this

paper for solving FS issues. The two proposed variants are (1) the binary whale optimization algorithm using S-shaped transfer function (bWOA-S) and (2) the binary whale optimization algorithm using V-shaped transfer function (bWOA-V). In both approaches, the accuracy of K-NN classifier [58] is used as an objective function that must be maximized. K-NN with leave-one-out cross-validation (LOOCV) based on Euclidean distance is also used to investigate the performance of the compared algorithms. The experiments results were evaluated on 24 datasets from UCI repository [59]. The results of the two proposed algorithms were evaluated versus different well-known algorithms famous in this domain, namely (1) particle swarm optimizer (PSO) [60], (2) three versions of binary ant lion (bALO1), bALO2, and bALO3) [51], (3) binary gray wolf Optimizer bGWO [50], (4) binary dragonfly [61] and (5) the original WOA. The reason behind choosing such algorithms is that PSO, one of the most famous and well-know algorithms, as well as bALO, bGWO, and bDA, are recent algorithms whose performance has been proved to be significant. Hence, we have implemented the compared algorithms using the original studies and then generated new results using these methods under the same circumstances. The experimental results revealed that bWOA-S and bWOA-V achieved higher classification accuracy with better feature reduction than the compared algorithms.

Therefore, the merits of the proposed algorithms versus the previous algorithms is illustrated by the following two aspects. First, bWOA-S and bWOA-V confirms not only feature reduction, but also the selection of relevant features. Second, bWOA-S and bWOA-V utilize the wrapper methods search technique for selecting prominent features, and hence the idea of these rules is based mainly on high classification accuracy regardless of a large number of selected features. The purpose of wrapper method is used to maintain an efficient balance between exploitation and exploration, so correct information of the features is provided [62]. Thus, bWOA-S and bWOA-V achieve a strong search capability that helps to select a minimum number of features as a subset from the most significant features pool.

The rest of the paper is organized as follows: Section 2 briefly introduces the WOA. Section 3, describes the two binary versions of whale optimization algorithm (bWOA), namely bWOA-S and bWOA-V, for feature selection. Section 4, discusses the empirical results for bWOA-S and bWOA-V. Eventually, conclusions and future work are drawn in Section 5.

2. Whale Optimization Algorithm

In [26], Mirjalili et al. introduced the whale optimization algorithm (WOA), based on the behaviour of whales. The special hunting method is considered the most interesting behaviour of humpack whales. This hunting technique is called bubble-net feeding. In the classical WOA, the solution of the current best candidate is set as close to either the optimum or the target prey. The other whales will update their position towards the best. Mathematically, the WOA mimics the collective movements as follows

$$\vec{D} = |\vec{C} \cdot \vec{X}^*(t) - \vec{X}(t)| \tag{1}$$

$$X(\vec{t+1}) = \vec{X}^*(t+1) - \vec{A} \cdot \vec{D} \tag{2}$$

where t refers to the current number of iterations, X refers to the position vector, X^* is the best solution position vector. C and A are coefficient vectors and can be calculated from the following equations

$$\vec{A} = 2 \cdot \vec{a} \cdot \vec{r} - \vec{a} \tag{3}$$

$$\vec{C} = 2 \cdot \vec{r} \tag{4}$$

where r belongs to the interval [0, 1] and a decreases linearly through the iterations from 2 to 0. WOA has two different phases: exploitation (Intensification) and exploration (diversification). In the diversification phase, the agents are moved for exploring or searching different search space regions, while in the intensification phase, the agents move in order to locally enhance the current solutions.

The intensification phase: the intensification phase is divided into two processes: the first one is the shrinking encircling technique which can be obtained by reducing a values using Equation (4). Note that a is a stochastic value in the interval $[-a, a]$. The second phase is the spiral updating position in which the distance between the whale and the prey is calculated. To model a spiral movement, the following equation is used in order to mimic the movement of the helix-shaped.

$$\vec{X}(t+1) = \vec{D}^{l} e^{bl} \cdot \cos(2\pi l)) + \vec{X}^{*}(t) \tag{5}$$

From Equation (5), l is a randomly chosen value between $[-1, 1]$ where b is a fixed. A 50% probability is used for choosing either the spiral model or shrinking encircling mechanism, as assumed. Consequently, the mathematical model is established as follows

$$\vec{X}(t+1) = \begin{cases} \vec{X^{*}}(t) - \vec{A} \cdot \vec{D} & if p < 0.5 \\ \vec{D^{l}}^{bl} \cdot \cos(2\pi l) + \vec{X^{*}}(t) & if p \geq 0.5 \end{cases} \tag{6}$$

where p is a random number in a uniform distribution.

The exploration phase: In the exploration phase, A used random values within $1 \prec A \prec -1$ to force the agent to move away from this location mathematically, formulated as in Equation (7).

$$\vec{D} = |\vec{C} \cdot \vec{X}_{rand} - \vec{X}| \tag{7}$$

$$\vec{X}(t+1) = X_{rand} - \vec{A} \cdot \vec{D} \tag{8}$$

3. Binary Whale Optimization Algorithm

In the classical WOA, whales move inside the continous search space in order to modify their positions, and this is called the continuous space. However, to solve FS issues, the solutions are limited to only $\{0, 1\}$ values. In order to be able to solve feature selection problems, the continuous (free position) must be converted to their corresponding binary solutions. Therefore, two binary versions from WOA are introduced to investigate problems like FS and achieve superior results. The conversion is performed by applying specific transfer functions, either the S-shaped function or V-shaped function in each dimension [63]. Transfer functions show the probability of converting the position vectors' from 0 to 1 and vice versa, i.e., force the search agents to move in a binary space. Figure 1 demonstrates the flow chart of the binary WOA version. Algorithm 1 shows the pseudo code of the proposed bWOA-S and bWOA-V versions.

3.1. Approach 1: Proposed bWOA-S

The common S-shaped (Sigmoid) function is used in this version. The S-shaped function is updating, as shown in Equation (11). Figure 2 illustrates the mathematical curve of the Sigmoid function.

3.2. Approach 2: Proposed bWOA-V

In this version, the hyperbolic tan function is applied. It is a common example of V-shaped functions and is given in Equations (9) and (10).

$$y^{k} = |tanhx^{k}| \tag{9}$$

$$X_{i}^{d} = \begin{cases} sel_{d}^{t} & if rand < S(x_{i}^{k}(t+1)) \\ org_{d}^{t} & otherwise \end{cases} \tag{10}$$

$$y^{k} = \frac{1}{1 + e^{-x_{i}^{k}(t)}} \tag{11}$$

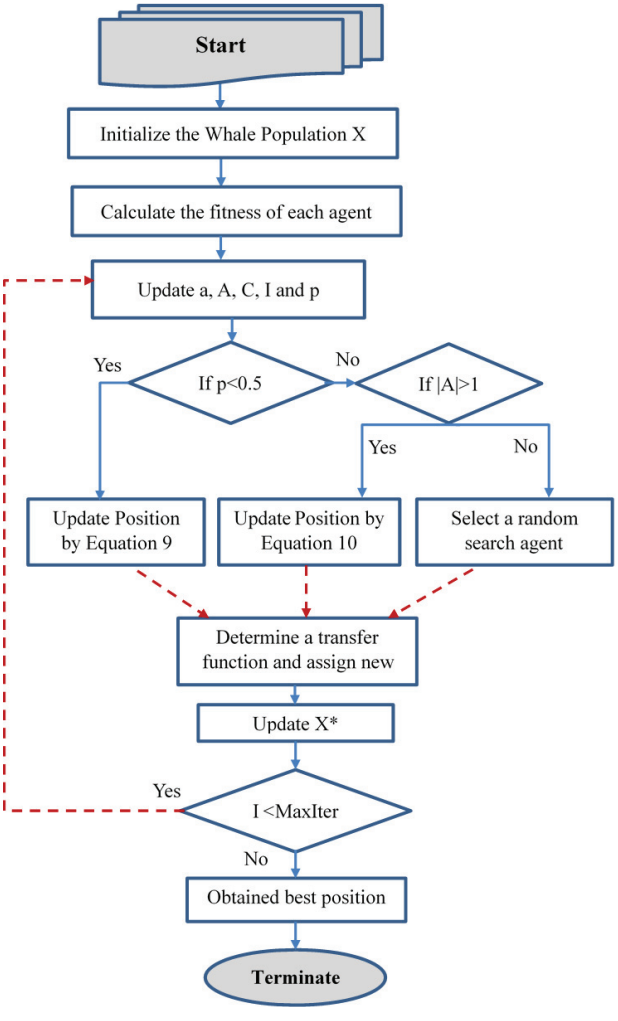


Figure 1. Binary whale optimization algorithm flowchart.

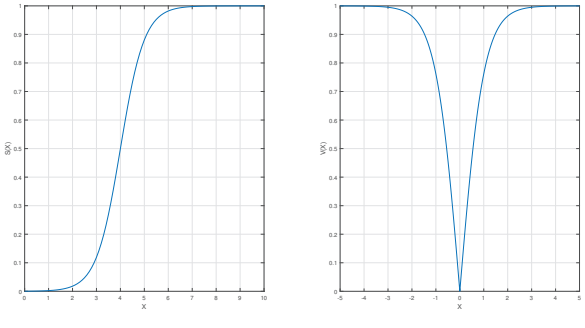


Figure 2. S-shaped and V-shaped transfer functions.

Algorithm 1 Pseudo code of bWOA-S & bWOA-V

Input: n whales number in the population.
$MaxIter$ maximum iteration number.
Output: position of the optimal whale.
Initialize a and n.
Calculate X^*.
while current iter $<$ maximum iteration number **do**
  for Each Whale **do**
    Calculate a; A, C, p and l.
    if $p < 0.5$ **then**
      if $(|A| < 1)$ **then**
        Update the position of whale using Equation (2).
      else$(|A| \geq 1)$
        Choose a random search agent (X_{rand})
        Update the position of whale using (8).
      end if
    else$(p \geq 0.5)$
      UUpdate the position of whale using (5).
    end if
    Update $\vec{X}(t+1)$ using Equation (11) or (9)
  end for
  Update $X*$ if there is a better solution.
  $t++$
end while

3.3. bWOA-S and bWOA-V for Feature Selection

Two binary variants of whale optimization algorithm, called bWOA-S and bWOA-V, are employed for solving the FS problem. For a feature vector size, if N is the number of different features, then the combination number would be 2^N, which is a huge feature number to search exhaustively. Under such a situation, the proposed bWOA-S and bWOA-V algorithms are used in an adaptive feature space search and provide the best combination of features. This combination is obtained by achieving the maximum classification accuracy and the minimum selected features number. The following Equation (12) shows the fitness function accompanied by the two proposed versions to evaluate individual whale positions.

$$F = \alpha\gamma_R(D) + \beta\frac{|C-R|}{|C|} \tag{12}$$

where F refers to Fitness function, R refers to the length of the selected feature subset, C refers to the total features number, $\gamma_R(D)$ refers to the classification accuracy of the condition attribute set R, α and β are two arguments that are symmetric to the subset length and the accuracy of the classification, and can be calculated as $\alpha \in [0,1]$ and $\beta = 1-\alpha$. This will lead to the fitness function that achieves the maximum classification accuracy. Equation (12) can be converted to a minimization problem based on

the error rate of classification and selected features. Thus, the obtained minimization problem can be calculated as in Equation (13)

$$F = \alpha E_R(D) + \beta \frac{|R|}{|C|} \tag{13}$$

where F refers to Fitness function, $E_R(D)$ is the classification error rate. According to the wrapper methods characteristic in FS, the classifier was employed as an FS guide. In this study, K-NN classifier is used. Therefore, K-NN is applied to ensure that the selected features are the most relevant ones. However, bWOA is the search method that tries to explore the feature space in order to maximize the feature evaluation criteria, as shown in Equation (13).

4. Experimental Results and Discussion

The two proposed bWOA-S and bWOA-V methods are compared with a group of existing algorithms, including the PSO, three variants of binary ant lion (bALO1, bALO2, and bALO3), and the original WOA. Table 1 reports the parameter settings for the cometitior algorithms. In order to provide a fair comparison, three initialization scenarios are used and the experimental results are performed using 24 different datasets from the UCI repository.

Table 1. Parameter setting.

Parameter	Value
No of search agents	8
No of iterations	70
Problem dimension	No. of features in the data
Data Search domain	[0, 1]
No. repetitions of runs	20
Inertia factor of PSO	0.1
Individual-best acceleration factor of PSO	0.1
α Parameter in the fitness function	0.99
β Parameter in the fitness function	0.01

4.1. Data Acquisition

Table 2 summarizes the 24 datasets from the UCI machine learning repository [59] that were used in the experiments. The datasets were selected with different instances and attribute numbers to represent various kinds of issue (small, medium and large). In each repository, the instances are divided randomly into three different subsets, namely training, testing, and validation subsets. The proposed algorithms were tested over three gene expression datasets of colon cancer, lymphoma and the leukemia [64–66]. The K-NN is used in the experimental tests using the trial and error method, and 5 is the best choice of K. Meanwhile, every position of whale produces one attribute subset through the training process. The training set is used to test and evaluate the performance of the K-NN classifier in the validation subset throughout the optimization process. The bWOA is employed to simultaneously guide the FS process.

Table 2. List of datasets used in the experiments results.

No.	Name	Features	Samples
1	Breastcancer	9	699
2	Tic-tac-toe	9	958
3	Zoo	16	101
4	WineEW	13	178
5	SpectEW	22	267
6	SonarEW	60	208
7	IonosphereEW	34	351
8	HeartEW	13	270
9	CongressEW	16	435
10	KrvskpEW	36	3196
11	WaveformEW	40	5000
12	Exactly	13	1000
13	Exactly 2	13	1000
14	M-of-N	13	1000
15	vote	16	300
16	BreastEW	30	569
17	Semeion	265	1593
18	Clean 1	166	476
19	Clean 2	166	6598
20	Lymphography	18	148
21	PenghungEW	325	73
22	Colon	2000	62
23	lymphoma	96	4026
24	Leukemia	7129	72

4.2. Evaluation Criteria

Each algorithm carried out 20 independent runs with a random initial positioning of the search agents. Repeated runs were used to test the capability of the convergence. Eight well-known and common measures are recorded in order to investigate the algorithms performance in a comparative way. Such metrics are listed as follows:

- Best: The minimum (or best for a minimization problem) fitness function value obtained at different independent runs, as depicted in Equation (14).

$$Best = Min_{i=1}^{M} g_*^i \tag{14}$$

- Worst: The maximum (or worst for a minimization) fitness function value obtained at different independent operations, as shown in Equation (15).

$$Worst = Max_{i=1}^{M} g_*^i \tag{15}$$

- Mean: Average calculation performance of the optimization algorithm applied M times, as shown in Equation (16).

$$Mean = \frac{1}{M} \sum_{i=1}^{M} g_*^i \tag{16}$$

  where g_*^i is the optimal solution obtained in the i-th operation;
- Standard deviation (Std) can be calculated from the following Equation (17).

$$Std = \sqrt{\frac{1}{M} \sum (g_*^i - Mean)^2} \tag{17}$$

- Average classification accuracy: Investigates the accuracy of the classifier and can be calculated by Equation (18).

$$AvgeragePerformance = \frac{1}{M}\sum_{j=1}^{M}\frac{1}{N}\sum_{i=1}^{N} Match(C_i, L_i) \tag{18}$$

  where C_i refers to classifier output for instance i; N refers to the instance number in the test set; and L_i refers to the reference class corresponding to instance i;
- Average selection size (Avg-Selection) measures the average reduction in selected features from all feature sets and is calculated by Equation (19)

$$AvgerageSelectionSize = \frac{1}{M}\sum_{i=1}^{M}\frac{size(g_*^i)}{N_t} \tag{19}$$

  where N_t is the total number of features in the original dataset;
- Average execution time (Avg-Time) measures the average execution time in milliseconds for all comparison optimization algorithms to obtain the results over the different runs and calculated by Equation (20)

$$R_a = \frac{1}{M}\sum_{i=1}^{M} RunT_{a,i} \tag{20}$$

  where M refers to the run number for the optimizer a, and $RunT_{a,i}$ is the computational time for optimizer a in milliseconds at run number i;
- Wilcoxon rank sum test (Wilcoxon): a non-parametric test called Wilcoxon Rank Sum (WRS) [67]. The test gives ranks to all the scores in one group, and after that the ranks of each group are added. The rank-sum test is often described as the non-parametric version of the t test for two independent groups.

The two proposed versions of whale optimization algorithm (bWOA-S and bWOA-V) are compared with three common algorithms that are famous in this domain. Four different initialization methods/techniques are used to guarantee the two proposed algorithms' ability to converge from different initial positions. These methods are: (1) a large initialization is expected to evaluate the capability of locally searching a given algorithm, as the search agents' positions are commonly close to the optimal solution; (2) a small initialization method is expected to evaluate the ability of a given algorithm to use global searching as the initial search; (3) mixed initialization is the case in which some search agents are close enough to the optimal solution, whereas the other search agents are apart. It will provide diversity of the population frequently. since the search agents are expected to be apart from each other. (4) random initialization.

4.3. Performance on Small Initialization

The statistical average fitness values of the different datasets obtained from the compared algorithms using the small initialization methods are shown in Table 3. Table 4 shows average classification accuracy on the test data of the compared algorithms using small initialization methods. From these tables, we can conclude that both bWOA-S and bWOA-V achieve better results compared with other algorithms.

4.4. Performance on Large Initialization

The statistical average fitness values of the different datasets obtained from the compared algorithms using the large initialization methods are shown in Table 5. Table 6 shows average classification accuracy of the test data of the compared algorithms using small initialization methods. From these tables, we can conclude that when using large initialization methods, both bWOA-S and bWOA-V achieve better results compared with other algorithms.

4.5. Performance on Mixed Initialization

The statistical average fitness values on the different datasets obtained from the compared algorithms using the large initialization methods are shown in Table 7. Table 8 shows average classification accuracy of the test data of the compared algorithms using small initialization methods. As is notable from this table, we can conclude that both bWOA-S and bWOA-V achieve better results compared with other algorithms.

Table 3. Statistical mean fitness measure on the different datasets calculated for the compared algorithms using small initialization.

No.	WOA	bWOA-S	bWOA-v	BALO1	BALO2	BALO3	PSO	bGWO	bDA
1	0.061	0.049	0.051	0.079	0.095	0.088	0.060	0.035	0.031
2	0.327	0.224	0.313	0.345	0.352	0.334	0.333	0.243	0.210
3	0.247	0.133	0.220	0.411	0.395	0.416	0.249	0.127	0.058
4	0.933	0.908	0.937	0.955	0.960	0.953	0.926	0.880	0.877
5	0.345	0.295	0.340	0.351	0.391	0.375	0.362	0.276	0.253
6	0.337	0.203	0.315	0.374	0.372	0.369	0.303	0.154	0.188
7	0.137	0.123	0.131	0.175	0.177	0.184	0.141	0.098	0.125
8	0.297	0.251	0.273	0.294	0.302	0.288	0.282	0.195	0.169
9	0.381	0.361	0.379	0.391	0.397	0.394	0.402	0.354	0.338
10	0.391	0.081	0.375	0.421	0.418	0.419	0.421	0.079	0.052
11	0.436	0.196	0.437	0.499	0.498	0.517	0.432	0.181	0.187
12	0.322	0.297	0.337	0.347	0.332	0.334	0.314	0.314	0.208
13	0.245	0.244	0.239	0.237	0.264	0.240	0.243	0.244	0.237
14	0.291	0.135	0.299	0.359	0.351	0.352	0.289	0.133	0.075
15	0.125	0.068	0.140	0.151	0.155	0.174	0.130	0.062	0.054
16	0.051	0.047	0.059	0.087	0.084	0.083	0.051	0.038	0.030
17	0.097	0.035	0.097	0.095	0.094	0.096	0.099	0.025	0.033
18	0.298	0.150	0.298	0.357	0.375	0.367	0.294	0.110	0.141
19	0.087	0.044	0.087	0.128	0.131	0.134	0.086	0.035	0.043
20	0.294	0.203	0.275	0.376	0.317	0.379	0.309	0.183	0.165
21	0.461	0.181	0.444	0.614	0.602	0.606	0.446	0.148	0.176

Table 4. Average classification accuracy for the compared algorithms on the different datasets using small initialization.

No.	WOA	bWOA-S	bWOA-v	BALO1	BALO2	BALO3	PSO	bGWO	bDA
1	0.863	0.648	0.745	0.834	0.814	0.842	0.867	0.966	0.758
2	0.652	0.781	0.670	0.598	0.599	0.584	0.620	0.743	0.685
3	0.740	0.843	0.770	0.457	0.471	0.442	0.588	0.862	0.817
4	0.041	0.057	0.026	0.014	0.011	0.017	0.033	0.088	0.033
5	0.624	0.663	0.606	0.566	0.557	0.550	0.583	0.705	0.640
6	0.632	0.712	0.658	0.547	0.548	0.549	0.609	0.832	0.696
7	0.845	0.835	0.838	0.780	0.779	0.761	0.820	0.890	0.828
8	0.674	0.645	0.632	0.602	0.592	0.604	0.653	0.793	0.658
9	0.585	0.584	0.587	0.557	0.540	0.572	0.565	0.629	0.584
10	0.586	0.919	0.606	0.517	0.519	0.519	0.545	0.916	0.782
11	0.556	0.804	0.552	0.398	0.402	0.392	0.392	0.817	0.742
12	0.635	0.668	0.618	0.588	0.622	0.619	0.656	0.656	0.640
13	0.725	0.722	0.703	0.744	0.692	0.704	0.724	0.728	0.710
14	0.699	0.845	0.845	0.720	0.723	0.708	0.814	0.932	0.873
15	0.864	0.915	0.838	0.720	0.723	0.708	0.814	0.932	0.873
16	0.899	0.694	0.724	0.808	0.821	0.833	0.893	0.963	0.780
17	0.897	0.964	0.890	0.876	0.902	0.903	0.898	0.971	0.956
18	0.685	0.815	0.674	0.593	0.582	0.589	0.641	0.875	0.796
19	0.909	0.957	0.908	0.847	0.848	0.842	0.884	0.965	0.952
20	0.674	0.734	0.654	0.513	0.553	0.523	0.616	0.799	0.706
21	0.491	0.748	0.493	0.285	0.295	0.300	0.415	0.809	0.729

4.6. Discussion

Figure 3 shows the effect of the initialization method on the different optimizers applied over the selected datasets. The proposed bWOA-S and bWOA-V can reach the global optimal solution in

almost half of the datasets, compared to the algorithms in all initialization methods. The limited search space in the case of binary algorithms explains the enhanced performance due to the balance between global and local searching. The balance between local and global searching assists the optimization algorithm to avoid early convergence and local optimal values. The small initialization keeps away the initial search agents from the optimal solution; however, in the large initialization, the search agents are closest to the optimal solution, although they have low diversity. While the mixed initialization method improves the performance of all compared algorithms, the two proposed algorithms are superior even in a high-dimensional dataset as in Table 9.

Table 5. Statistical mean fitness measure calculated on the different datasets for the compared algorithms using large initialization.

No.	WOA	bWOA-S	bWOA-v	BALO1	BALO2	BALO3	PSO	bGWO	bDA
1	0.133	0.127	0.164	0.183	0.146	0.223	0.160	0.036	0.032
2	0.215	0.207	0.209	0.241	0.248	0.243	0.204	0.211	0.209
3	0.149	0.138	0.139	0.168	0.129	0.182	0.171	0.101	0.076
4	0.928	0.928	0.929	0.938	0.937	0.924	0.925	0.907	0.882
5	0.316	0.312	0.314	0.322	0.320	0.312	0.314	0.303	0.249
6	0.303	0.289	0.293	0.273	0.298	0.288	0.277	0.258	0.197
7	0.168	0.163	0.180	0.162	0.177	0.166	0.160	0.150	0.127
8	0.349	0.337	0.349	0.341	0.358	0.346	0.345	0.288	0.171
9	0.400	0.403	0.390	0.403	0.403	0.388	0.397	0.375	0.343
10	0.069	0.073	0.072	0.073	0.071	0.073	0.069	0.067	0.051
11	0.193	0.192	0.192	0.196	0.193	0.191	0.188	0.189	0.187
12	0.303	0.309	0.312	0.305	0.305	0.304	0.302	0.305	0.207
13	0.259	0.259	0.260	0.260	0.266	0.264	0.258	0.256	0.241
14	0.138	0.131	0.138	0.143	0.137	0.133	0.121	0.121	0.068
15	0.087	0.090	0.086	0.089	0.093	0.094	0.086	0.084	0.053
16	0.217	0.220	0.156	0.108	0.155	0.205	0.200	0.043	0.030
17	0.044	0.043	0.044	0.043	0.042	0.045	0.046	0.036	0.033
18	0.187	0.186	0.189	0.182	0.195	0.190	0.189	0.170	0.138
19	0.052	0.052	0.053	0.052	0.051	0.052	0.051	0.049	0.043
20	0.238	0.232	0.222	0.248	0.235	0.233	0.234	0.228	0.147
21	0.260	0.246	0.273	0.274	0.262	0.273	0.232	0.227	0.183

Table 6. Average classification accuracy on the different datasets for the compared algorithms using large initialization.

No.	WOA	bWOA-S	bWOA-v	BALO1	BALO2	BALO3	PSO	bGWO	bDA
1	0.616	0.619	0.615	0.679	0.693	0.666	0.748	0.959	0.780
2	0.792	0.799	0.798	0.740	0.738	0.742	0.748	0.760	0.668
3	0.833	0.839	0.832	0.811	0.847	0.798	0.817	0.890	0.787
4	0.059	0.056	0.054	0.048	0.050	0.062	0.060	0.084	0.033
5	0.664	0.670	0.668	0.663	0.668	0.674	0.668	0.688	0.643
6	0.692	0.703	0.698	0.719	0.696	0.705	0.720	0.741	0.704
7	0.830	0.836	0.819	0.838	0.821	0.832	0.839	0.852	0.819
8	0.645	0.654	0.637	0.648	0.630	0.639	0.642	0.697	0.653
9	0.593	0.583	0.598	0.581	0.580	0.593	0.586	0.620	0.589
10	0.934	0.930	0.932	0.918	0.925	0.923	0.931	0.939	0.777
11	0.810	0.808	0.810	0.804	0.807	0.810	0.813	0.815	0.740
12	0.693	0.683	0.685	0.680	0.680	0.679	0.684	0.689	0.648
13	0.740	0.741	0.741	0.728	0.723	0.724	0.734	0.737	0.712
14	0.861	0.865	0.862	0.831	0.833	0.834	0.856	0.866	0.721
15	0.907	0.908	0.905	0.907	0.903	0.901	0.906	0.917	0.881
16	0.612	0.610	0.613	0.715	0.697	0.656	0.714	0.938	0.766
17	0.963	0.964	0.963	0.964	0.965	0.962	0.962	0.971	0.958
18	0.814	0.818	0.812	0.820	0.807	0.812	0.812	0.834	0.807
19	0.956	0.956	0.955	0.955	0.957	0.956	0.956	0.959	0.953
20	0.742	0.754	0.762	0.736	0.746	0.752	0.745	0.770	0.717
21	0.742	0.755	0.731	0.729	0.742	0.730	0.769	0.773	0.731

The standard deviation in the obtained fitness values on the different datasets for the compared algorithms averaged over the initialization methods is given in Table 10. As shown in this table, the proposed bWOA-V can reach the optimal solution better than compared algorithms, regardless of the initialization used.

With regard to the time consumption for optimization of these 11 test datasets, Table 11 presents the results of the average time obtained by the two proposed versions and other compared algorithms with 20 independent runs. As can be concluded from Table 11, bWOA-V ranks first among the algorithms. bWOA-S ranks fifth, but it is better than PSO and bALO, as it significantly outperforms the other compared algorithms with a little more time consumption.

Table 7. Statistical mean fitness measure calculated on the different datasets for the compared algorithms using mixed initialization.

No.	WOA	bWOA-S	bWOA-v	BALO1	BALO2	BALO3	PSO	bGWO	bDA
1	0.054	0.052	0.079	0.100	0.099	0.076	0.031	0.035	0.032
2	0.220	0.207	0.215	0.245	0.252	0.246	0.204	0.215	0.209
3	0.153	0.148	0.120	0.183	0.146	0.141	0.078	0.096	0.071
4	0.925	0.928	0.910	0.935	0.938	0.938	0.884	0.903	0.882
5	0.313	0.307	0.289	0.319	0.321	0.312	0.242	0.280	0.255
6	0.304	0.286	0.254	0.278	0.298	0.285	0.168	0.235	0.194
7	0.159	0.158	0.152	0.156	0.169	0.165	0.113	0.141	0.124
8	0.328	0.308	0.259	0.319	0.324	0.308	0.158	0.233	0.167
9	0.389	0.380	0.372	0.393	0.397	0.384	0.337	0.359	0.341
10	0.071	0.074	0.081	0.074	0.072	0.074	0.040	0.061	0.053
11	0.193	0.193	0.195	0.198	0.195	0.193	0.182	0.187	0.188
12	0.303	0.308	0.301	0.301	0.307	0.308	0.151	0.272	0.226
13	0.241	0.244	0.252	0.237	0.244	0.253	0.238	0.244	0.243
14	0.139	0.133	0.155	0.151	0.150	0.136	0.022	0.112	0.072
15	0.084	0.084	0.081	0.089	0.090	0.085	0.048	0.069	0.052
16	0.081	0.058	0.062	0.086	0.088	0.086	0.033	0.057	0.031
17	0.044	0.043	0.037	0.043	0.043	0.044	0.032	0.034	0.030
18	0.191	0.187	0.176	0.184	0.192	0.197	0.136	0.158	0.149
19	0.052	0.052	0.049	0.051	0.052	0.052	0.041	0.044	0.042
20	0.235	0.230	0.223	0.258	0.243	0.237	0.138	0.211	0.160
21	0.260	0.244	0.242	0.276	0.262	0.274	0.149	0.217	0.180

Table 8. Average classification accuracy on the different datasets for the compared algorithms using mixed initialization.

No.	WOA	bWOA-S	bWOA-v	BALO1	BALO2	BALO3	PSO	bGWO	bDA
1	0.785	0.619	0.628	0.740	0.725	0.726	0.802	0.962	0.789
2	0.787	0.799	0.786	0.686	0.681	0.686	0.720	0.764	0.673
3	0.841	0.839	0.822	0.656	0.706	0.680	0.789	0.900	0.779
4	0.065	0.056	0.053	0.039	0.033	0.031	0.039	0.086	0.031
5	0.678	0.670	0.664	0.635	0.623	0.625	0.656	0.707	0.649
6	0.698	0.703	0.703	0.645	0.639	0.647	0.721	0.765	0.705
7	0.835	0.836	0.831	0.819	0.803	0.802	0.835	0.860	0.827
8	0.656	0.654	0.652	0.625	0.621	0.623	0.668	0.751	0.652
9	0.598	0.582	0.595	0.573	0.559	0.577	0.589	0.631	0.571
10	0.936	0.930	0.918	0.766	0.765	0.757	0.794	0.943	0.754
11	0.812	0.808	0.804	0.642	0.649	0.647	0.763	0.816	0.747
12	0.687	0.683	0.691	0.644	0.656	0.648	0.664	0.706	0.642
13	0.738	0.740	0.735	0.733	0.711	0.703	0.723	0.735	0.712
14	0.865	0.865	0.833	0.734	0.732	0.744	0.761	0.883	0.728
15	0.915	0.908	0.900	0.829	0.823	0.829	0.884	0.930	0.866
16	0.761	0.610	0.615	0.730	0.744	0.727	0.810	0.944	0.769
17	0.964	0.964	0.965	0.924	0.939	0.925	0.956	0.972	0.959
18	0.815	0.818	0.803	0.729	0.720	0.724	0.806	0.845	0.791
19	0.956	0.956	0.955	0.908	0.910	0.911	0.953	0.962	0.952
20	0.756	0.755	0.749	0.639	0.672	0.659	0.705	0.786	0.709
21	0.744	0.755	0.725	0.553	0.568	0.563	0.765	0.781	0.730

Table 9. Results for high dimensional datasets.

Dataset	Accuracy	STDEV	Fitness			Time	SelSize
			Avg	Min	Max		
Colon							
WOA	0.67083	0.02710	0.52313	0.18933	0.33625	5.77346	0.52313
bWOA-S	0.66667	0.03066	0.45386	0.18940	0.31566	15.37727	0.45386
bWOA-V	0.66667	0.03003	0.49724	0.23179	0.35564	9.87549	0.49724
bALO1	0.62250	0.03513	0.46110	0.20995	0.35688	3.52489	0.46110
bALO2	0.62584	0.04386	0.47458	0.23059	0.37749	39.64500	0.47458
bALO3	0.62084	0.03544	0.49837	0.27183	0.35686	37.76940	0.49836
PSO	0.66084	0.02626	0.48793	0.16870	0.31424	3.52425	
bGWO1	0.79584	0.03536	0.35911	0.12644	0.27228	44.10091	0.35911
bDA	0.65167	0.02854	0.43856	0.16915	0.25231	6.72146	0.43856
Lymphoma							
WOA	0.42628	0.06076	0.47314	0.38451	0.72184	13.73907	0.47314
bWOA-S	0.35435	0.06035	0.44921	0.17422	0.71399	52.34015	0.44921
bWOA-V	0.39457	0.05754	0.49642	0.37169	0.80664	22.35106	0.49642
bALO1	0.41973	0.06194	0.51039	0.42877	0.73545	8.27976	0.510395
bALO2	0.39939	0.06230	0.44844	0.33482	0.74161	77.97951	0.44844
bALO3	0.39923	0.05861	0.48594	0.41489	0.76677	81.05818	0.48594
PSO	0.46635	0.05212	0.47878	0.18666	0.71151	7.31112	
bGWO1	0.48642	0.05491	0.28062	0.26272	0.71343	89.87190	0.28062
bDA	0.40717	0.03993	0.37595	0.32643	0.84836	16.47820	0.37595
Leukemia							
WOA	0.82353	0.08431	0.64732	0.15909	0.21848	30.54245	0.64732
bWOA-S	0.82353	0.08471	0.69674	0.07869	0.15941	85.80836	0.69674
bWOA-V	0.84647	0.07902	0.57925	0.09967	0.20281	45.17051	0.57925
bALO1	0.72500	0.08793	0.62512	0.14453	0.23919	14.68936	0.62511
bALO2	0.72471	0.09272	0.62429	0.15182	0.23920	171.695	0.62429
bALO3	0.73029	0.08913	0.62491	0.12273	0.23192	182.292	0.62491
PSO	0.85059	0.08055	0.80121	0.06281	0.1652	15.26511	
bGWO1	0.94588	0.07589	0.47347	0.02565	0.09169	205.829	0.47348
bDA	0.83706	0.07626	0.48777	0.02671	0.06319	31.56270	0.48777

Table 10. Standard deviation fitness function on the different datasets averaged for the compared algorithms over the three initialization methods.

No.	WOA	bWOA-S	bWOA-v	BALO1	BALO2	BALO3	PSO	bGWO	bDA
1	0.013	0.013	0.011	0.028	0.012	0.013	0.009	0.009	0.007
2	0.053	0.045	0.058	0.056	0.056	0.057	0.058	0.047	0.048
3	0.033	0.017	0.040	0.046	0.041	0.042	0.018	0.020	0.015
4	0.205	0.208	0.200	0.208	0.212	0.214	0.202	0.200	0.199
5	0.061	0.073	0.066	0.072	0.080	0.064	0.075	0.072	0.052
6	0.074	0.053	0.062	0.067	0.064	0.071	0.049	0.042	0.046
7	0.027	0.030	0.030	0.035	0.043	0.035	0.026	0.035	0.028
8	0.060	0.060	0.057	0.061	0.062	0.058	0.052	0.054	0.039
9	0.084	0.084	0.079	0.087	0.092	0.089	0.080	0.075	0.076
10	0.034	0.015	0.028	0.035	0.036	0.043	0.033	0.017	0.012
11	0.058	0.043	0.067	0.061	0.061	0.062	0.058	0.041	0.040
12	0.065	0.070	0.067	0.068	0.066	0.068	0.061	0.071	0.045
13	0.055	0.058	0.055	0.055	0.070	0.055	0.051	0.051	0.051
14	0.037	0.033	0.041	0.043	0.054	0.042	0.038	0.021	0.012
15	0.022	0.016	0.023	0.024	0.027	0.030	0.022	0.013	0.010
16	0.037	0.031	0.009	0.011	0.033	0.013	0.026	0.010	0.006
17	0.012	0.009	0.012	0.012	0.011	0.013	0.011	0.007	0.007
18	0.054	0.032	0.042	0.052	0.050	0.050	0.044	0.027	0.034
19	0.013	0.010	0.014	0.013	0.017	0.016	0.012	0.009	0.009
20	0.049	0.041	0.031	0.067	0.055	0.067	0.050	0.039	0.040
21	0.051	0.071	0.056	0.069	0.071	0.087	0.046	0.020	0.040

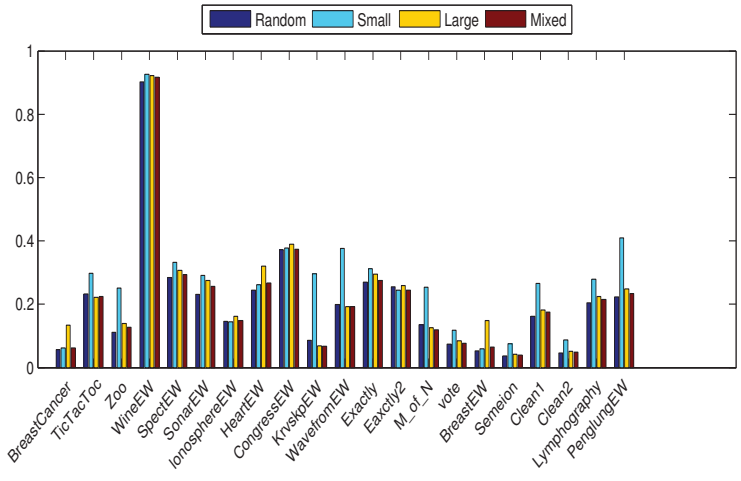


Figure 3. Statistical mean fitness averaged on the different datasets for the different optimizers using the different initializers.

Table 11. Average execution time in seconds on the different datasets for the compared algorithms averaged over the three initialization methods.

No.	WOA	bWOA-S	bWOA-v	BALO1	BALO2	BALO3	PSO	bGWO	bDA
1	4.722	4.243	4.467	4.896	4.703	4.42890	6.172	7.177	4.629
2	9.747	8.784	9.114	7.791	7.468	6.81349	8.253	10.149	6.720
3	3.712	3.460	3.741	4.019	4.010	3.89503	3.822	4.771	3.859
4	11.094	10.557	12.450	11.934	10.94404	10.779	11.804	14.946	11.225
5	3.725	3.364	4.072	4.158	4.195	3.92277	4.423	5.218	3.654
6	3.835	3.540	3.816	3.673	5.014	4.83652	4.316	5.756	3.599
7	4.139	4.220	4.376	4.030	4.978	4.73393	4.456	5.670	4.033
8	3.714	3.124	3.642	3.614	3.796	3.87177	4.029	5.339	3.616
9	4.353	3.719	4.680	4.130	4.502	4.66121	4.411	133	4.477
10	78.311	78.516	77.182	65.795	64.663	57.458	39.671	78.063	51.987
11	180	2122	3449	157	153	140	112	199	116
12	6.610	8.068	8.672	8.004	7.259	6.740	6.287	7.011	6.468
13	7.210	8.422	9.819	8.554	6.946	6.554	6.783	6.720	7.123
14	7.334	8.638	6.957	8.169	6.332	6.519	7.789	7.856	6.569
15	3.281	3.901	3.307	4.267	3.668	3.717	4.213	3.695	3.303
16	4.248	4.600	3.919	5.464	4.751	4.294	4.995	5.090	3.813
17	107	139	144	91.552	95.185	77.564	86.140	99.636	122
18	9.497	11.970	17.209	8.412	10.710	11.481	8.893	10.474	5.933
19	2672	1996	1733	985	1018	858	920	2053	1281
20	3.593	3.932	3.917	3.605	3.683	3.396	3.809	3.941	3.087
21	4.830	6.478	5.522	3.993	10.437	10.220	4.407	7.852	4.183

On the other hand, Tables 12 and 13 summarize the experimental results of the best and worst obtained fitness for the compared algorithms over 20 independent runs.

The mean selected features obtained from the compared algorithms are shown in Table 14.

Table 14 reports the ratio of mean selected features obtained from the compared algorithms. In Table 14, the performance of bWOA-V is superior in keeping its good classification accuracy by selecting a lower number of features.

This reveals the outstanding performance of bWOA-V in searching for both features' reduction and enhancing the optimization process.

Table 12. Best fitness function on the different datasets averaged for the compared algorithms over the three initialization methods.

No.	WOA	bWOA-S	bWOA-v	BALO1	BALO2	BALO3	PSO	bGWO	bDA
1	0.032	0.031	0.031	0.033	0.038	0.038	0.025	0.022	0.020
2	0.201	0.185	0.186	0.234	0.225	0.233	0.195	0.192	0.172
3	0.023	0.046	0.065	0.093	0.074	0.123	0.057	0.015	0.004
4	0.853	0.861	0.873	0.884	0.862	0.877	0.849	0.830	0.812
5	0.250	0.247	0.230	0.275	0.274	0.249	0.225	0.218	0.216
6	0.218	0.182	0.190	0.213	0.208	0.220	0.175	0.132	0.137
7	0.108	0.123	0.107	0.122	0.128	0.124	0.088	0.077	0.083
8	0.229	0.214	0.207	0.241	0.247	0.214	0.168	0.140	0.125
9	0.334	0.343	0.324	0.346	0.342	0.339	0.328	0.328	0.310
10	0.103	0.057	0.097	0.117	0.125	0.126	0.106	0.038	0.038
11	0.212	0.179	0.196	0.262	0.258	0.261	0.200	0.171	0.177
12	0.273	0.186	0.281	0.276	0.278	0.283	0.144	0.185	0.026
13	0.222	0.225	0.220	0.221	0.226	0.226	0.217	0.216	0.217
14	0.131	0.085	0.123	0.154	0.133	0.170	0.061	0.046	0.012
15	0.045	0.042	0.036	0.050	0.038	0.046	0.029	0.043	0.027
16	0.028	0.028	0.029	0.039	0.040	0.035	0.024	0.023	0.018
17	0.049	0.030	0.045	0.044	0.042	0.045	0.040	0.022	0.024
18	0.150	0.128	0.161	0.179	0.191	0.185	0.143	0.092	0.109
19	0.051	0.041	0.049	0.056	0.060	0.062	0.049	0.037	0.038
20	0.180	0.150	0.116	0.196	0.161	0.183	0.115	0.119	0.115
21	0.136	0.122	0.174	0.206	0.184	0.238	0.111	0.071	0.046

Table 13. Worst fitness function on the different datasets averaged for the compared algorithms over the three initialization methods.

No.	WOA	bWOA-S	bWOA-v	BALO1	BALO2	BALO3	PSO	bGWO	bDA
1	0.171	0.239	0.339	0.251	0.341	0.282	0.151	0.145	0.047
2	0.304	0.264	0.322	0.345	0.338	0.330	0.271	0.262	0.238
3	0.301	0.256	0.267	0.360	0.331	0.326	0.290	0.238	0.154
4	0.978	0.993	0.961	0.981	0.985	0.993	0.945	0.956	0.922
5	0.377	0.356	0.394	0.384	0.417	0.421	0.401	0.328	0.288
6	0.375	0.387	0.356	0.365	0.389	0.372	0.290	0.286	0.256
7	0.214	0.176	0.215	0.204	0.222	0.213	0.190	0.182	0.165
8	0.376	0.360	0.386	0.411	0.407	0.383	0.301	0.347	0.198
9	0.442	0.419	0.446	0.480	0.455	0.436	0.433	0.399	0.375
10	0.211	0.106	0.224	0.234	0.231	0.222	0.194	0.142	0.064
11	0.315	0.207	0.321	0.301	0.301	0.322	0.273	0.199	0.198
12	0.354	0.333	0.365	0.371	0.372	0.379	0.324	0.335	0.294
13	0.303	0.276	0.278	0.282	0.345	0.286	0.287	0.275	0.262
14	0.220	0.198	0.277	0.264	0.265	0.255	0.197	0.181	0.127
15	0.190	0.124	0.198	0.150	0.203	0.192	0.118	0.118	0.077
16	0.314	0.183	0.343	0.334	0.328	0.251	0.148	0.235	0.046
17	0.072	0.050	0.069	0.065	0.066	0.071	0.062	0.041	0.042
18	0.284	0.222	0.261	0.280	0.286	0.279	0.233	0.202	0.185
19	0.069	0.056	0.069	0.080	0.082	0.082	0.061	0.049	0.048
20	0.337	0.305	0.303	0.374	0.352	0.394	0.289	0.274	0.202
21	0.440	0.381	0.462	0.474	0.481	0.528	0.433	0.365	0.312

In order to compare each runs results, a non-parametric statistical called Wilcoxon's rank sum (WRS) test was carried out over the 11 UCI datasets at 5% significance level, and the *p*-values are given in Table 15. From this table, *p*-values for the bWOA-V are mostly less than 0.05, which proves that this algorithm's superiority is statistically significant. This means that bWOA-V exhibits a statistically superior performance compared to the other compared algorithms in the pair-wise Wilcoxon signed-ranks test.

Table 14. Average selection size on the different datasets averaged for the compared algorithms over the three initialization methods.

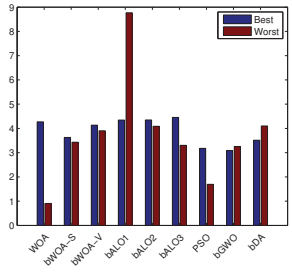
No.	WOA	bWOA-S	bWOA-v	BALO1	BALO2	BALO3	PSO	bGWO	bDA
1	0.60875	0.63875	0.56750	0.47500	0.50250	0.50875	0.636	0.63875	0.50625
2	0.77500	0.97083	0.75555	0.61806	0.63750	0.62083	0.520	0.79167	0.80417
3	0.66172	0.76328	0.60625	0.62031	0.61797	0.62500	0.609	0.59141	0.47109
4	0.62596	0.69904	0.58365	0.55865	0.56154	0.54231	0.643	0.58269	0.47019
5	0.64602	0.73920	0.59148	0.54432	0.59886	0.56989	0.568	0.62898	0.45966
6	0.64729	0.66396	0.55667	0.60563	0.60566	0.62396	0.520	0.62146	0.43854
7	0.60221	0.66875	0.59265	0.54522	0.55699	0.54081	0.564	0.61213	0.40625
8	0.55577	0.54519	0.54134	0.51731	0.45769	0.47885	0.611	0.57596	0.41730
9	0.53281	0.58438	0.54609	0.50859	0.52578	0.50469	0.427	0.62891	0.44219
10	0.70417	0.90347	0.67951	0.61909	0.62535	0.62361	0.578	0.76314	0.53368
11	0.73344	0.90500	0.70750	0.62656	0.63156	0.63062	0.750	0.79906	0.58656
12	0.64038	0.72693	0.69712	0.51635	0.54231	0.54231	0.475	0.62212	0.61827
13	0.49904	0.46731	0.61538	0.39423	0.40385	0.44615	0.475	0.42981	0.17885
14	0.72404	0.87884	0.69135	0.62212	0.60865	0.62115	0.695	0.76442	0.63462
15	0.66719	0.74609	0.60234	0.59141	0.56640	0.61016	0.520	0.61094	0.37813
16	0.57250	0.62375	0.60250	0.51875	0.49500	0.51000	0.552	0.60750	0.48875
17	0.66788	0.79953	0.59774	0.62183	0.62538	0.62363	0.856	0.64108	0.50028
18	0.69247	0.79488	0.58893	0.62146	0.61942	0.62387	0657	0.64932	0.48532
19	0.66822	0.77086	0.57515	0.62432	0.62402	0.62771	0.781	0.68577	0.48735
20	0.66250	0.72708	0.60069	0.60555	0.58958	0.59028	0.499	0.62569	0.50486
21	0.64835	0.71131	0.53630	0.62142	0.62111	0.62312	0.550	0.49126	0.47477

Table 15. The Wilcoxon test for the average fitness obtained by the compared algorithms.

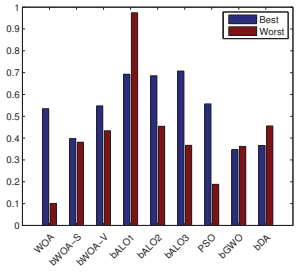
Algorithms	bWOA-S			bWOA-V		
	Small	Mixed	Large	Small	Mixed	Large
WOA	0.0606	0.4756	0.4201	0.4178	0.4352	0.5640
bALO1	0.0000	0.4006	0.4609	0.1191	0.2180	0.4480
bALO2	0.0038	0.2736	0.4248	0.0754	0.2036	0.5881
bALO3	0.0947	0.0596	0.6410	0.3404	0.0725	0.4672
bGWO	0.0589	0.0532	0.879	0.654	0.0587	0.0.300
bDA	0.0439	0.0298	0.1406	0.4892	0.0584	0.400

Moreover, Figure 4 outlines the best and worst acquired fitness function value averaged over all the datasets, using small, mixed and large initialization. Figure 5 shows the classification accuracy average. From these figures, it can be proven that the bWOA-V performs better than other compared algorithms, such as PSO and bALO, which confirms bWOA-V's searching capability, especially in the large initialization.

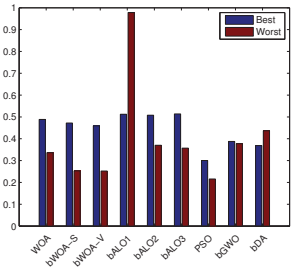
In order to show the merits of bWOA-S and bWOA-V qualitatively, Figures 6–8, show the boxplots results for the three initialization methods obtained by all compared algorithms. According to these figures, bWOA-S and bWOA-V have superiority since the boxplot of bWOA-S and bWOA-V are extremely narrow and located under the minima of PSO, bALO, and the original WOA. In summary, the qualitative results prove that the two proposed algorithms are able to provide remarkable convergence and coverage ability in solving FS problems. Another fact worth mentioning here is that the boxplots show that bALO and PSO algorithms provide poor performance.



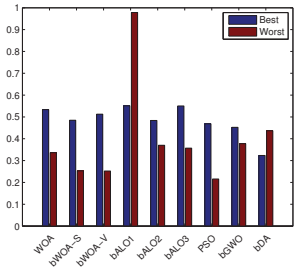
(**a**) Random initialization



(**b**) Small initialization

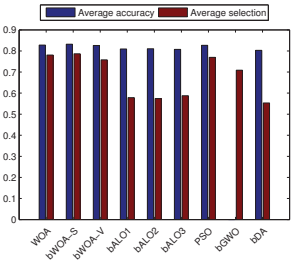


(**c**) Mixed initialization

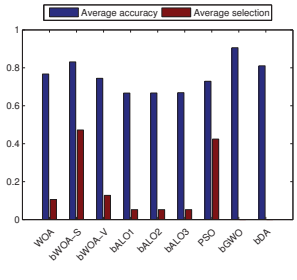


(**d**) Large initialization

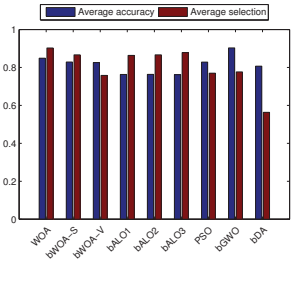
Figure 4. Best and worst fitness obtained for the compared algorithms on the different datasets averaged over the four initialization methods.



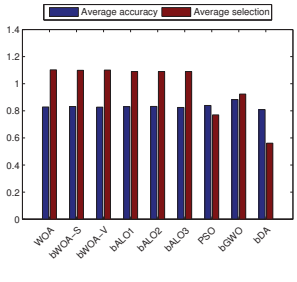
(**a**) Random initialization



(**b**) Small initialization



(**c**) Mixed initialization



(**d**) Large initialization

Figure 5. Average classification accuracy and average selection size obtained on the different datasets averaged for the compared algorithms over the three initialization methods.

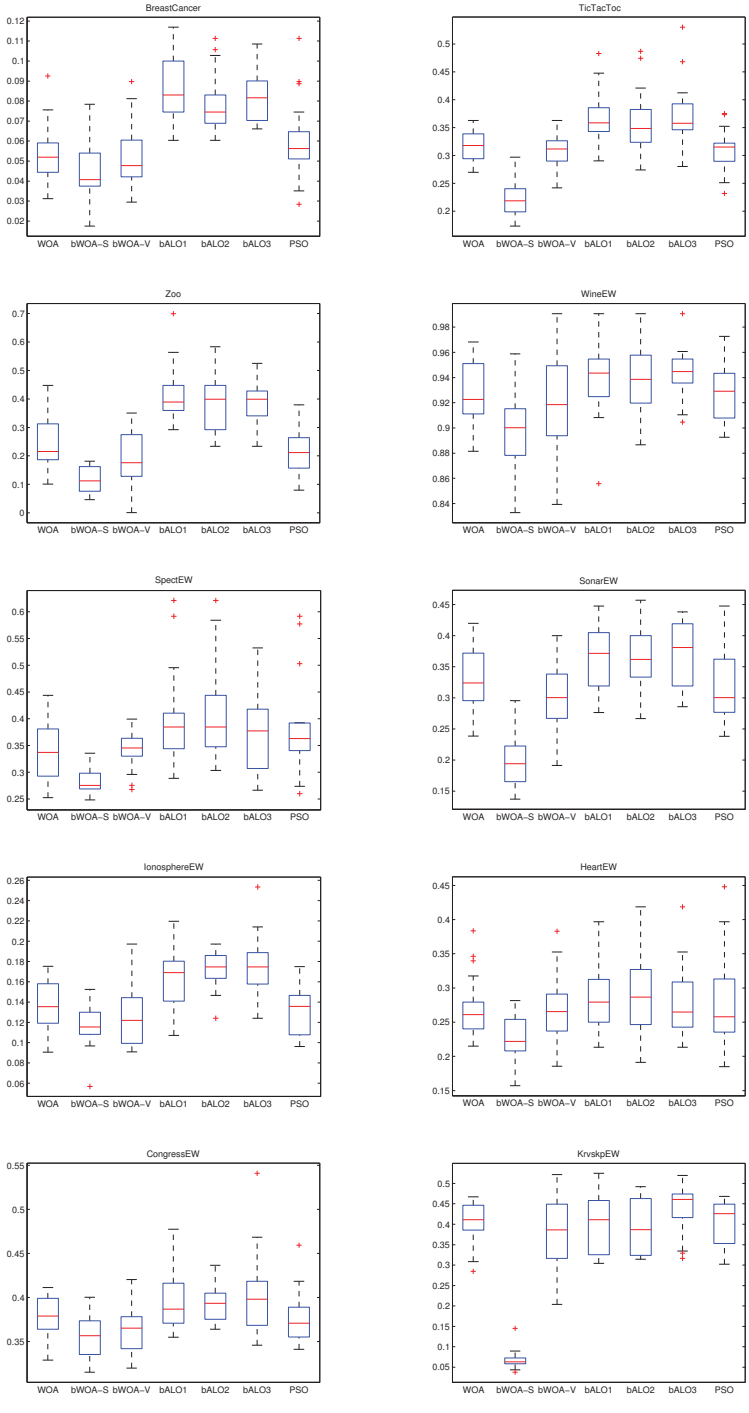


Figure 6. Small initialization boxplot for the compared algorithms on the different datasets.

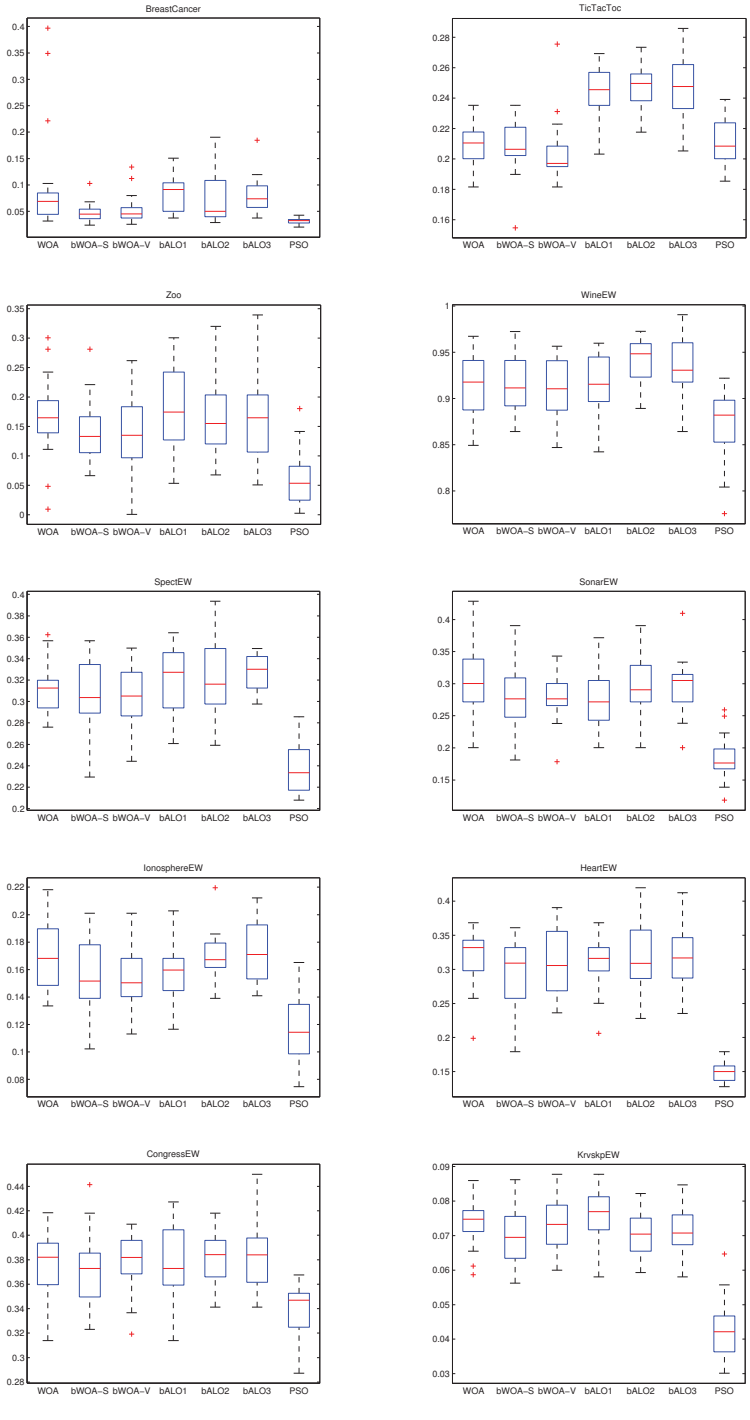


Figure 7. Mixed initialization boxplot for the compared algorithms on the different datasets.

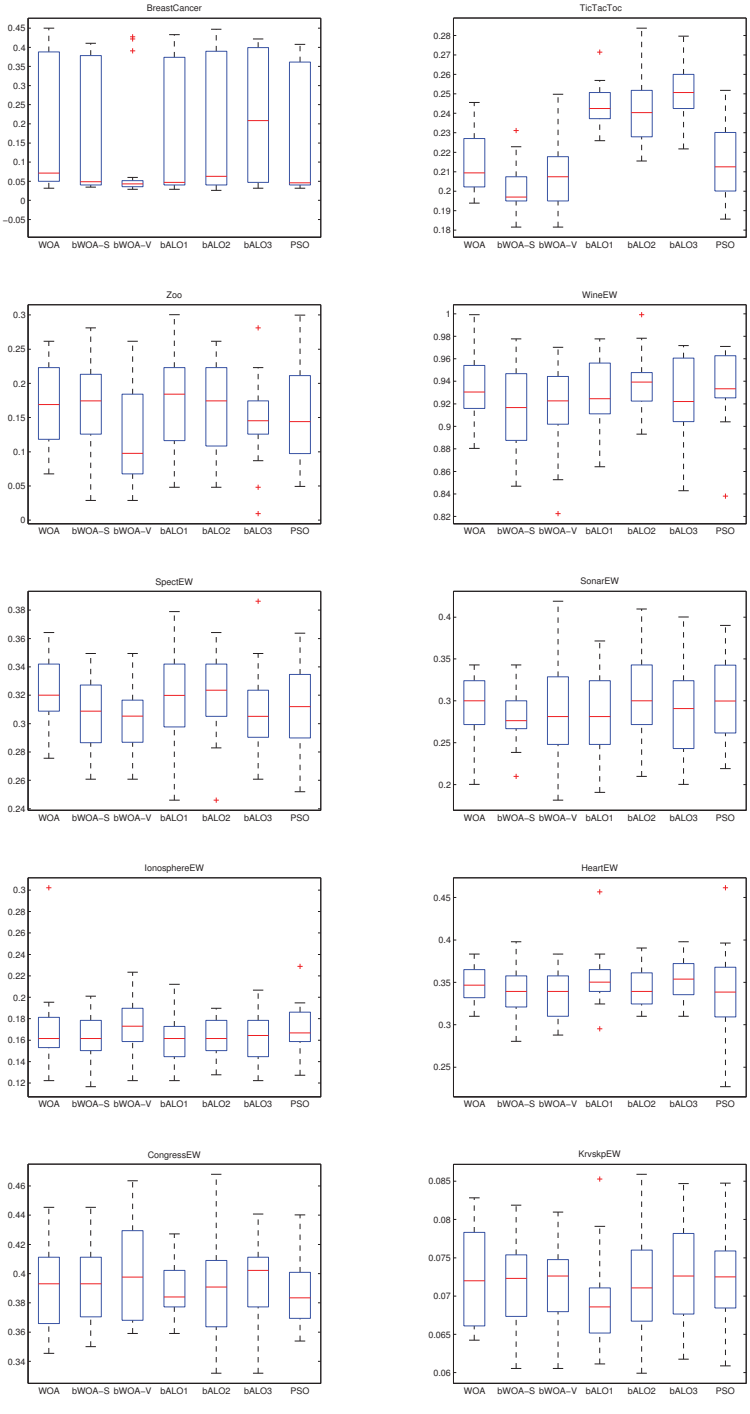


Figure 8. Large initialization boxplot for the compared algorithms on the different datasets.

5. Conclusions and Future Work

In this paper, two binary version of the original whale optimization algorithm (WOA), called bWOA-S and bWOA-V, have been proposed to solve the FS problem. To convert the original version of WOA to a binary version, S-shaped and V-shaped transfer functions are employed. In order to investigate the performance of the two proposed algorithms, the experiments employ 24 benchmark datasets from the UCI repository and eight evaluation criteria to assess different aspects of the compared algorithms.The experimental results revealed that the two proposed algorithms achieved superior results compared to the three well-known algorithms, namely PSO, bALO (three variants), and the original WOA. Furthermore, the results proved that bWOA-S and bWOA-V both achieved smallest number of selected features with best classification accuracy in a minimum time. In addition, the Wilcoxon's rank-sum nonparametric statistical test was carried out at 5% significance level to judge whether the results of the two proposed algorithms differ from the best results of the other compared algorithms in a statistically significant way. More specifically, the results proved that the bWOA-s and bWOA-V have merit among binary optimization algorithms. For future work, the two binary algorithms introduced here will be applied to high-dimensional real-world applications and will be used with more common classifiers such as SVM and ANN to verify the performance. The effects of different transfer functions on the performance of the two proposed algorithms are also worth investigating. This algorithm can be applied for many problems other than FS. We can also investigate a multi-objective version.

Author Contributions: A.G.H.: Software, Resources, Writing—original draft, editing. D.O.: Conceptualization, Data curation, Resources, Writing—review and editing. E.H.H.: Supervision, Methodology, Conceptualization, Formal analysis, Writing—review and editing. A.A.J.: Formal analysis, Writing—review and editing. X.Y.: Formal analysis, Writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external fundin.

Conflicts of Interest: The authors declare that there is no conflict of interest.

References

1. Yi, J.H.; Deb, S.; Dong, J.; Alavi, A.H.; Wang, G.G. An improved NSGA-III algorithm with adaptive mutation operator for Big Data optimization problems. *Future Gener. Comput. Syst.* **2018**, *88*, 571–585. [CrossRef]
2. Neggaz, N.; Houssein, E.H.; Hussain, K. An efficient henry gas solubility optimization for feature selection. *Expert Syst. Appl.* **2020**, *152*, 113364. [CrossRef]
3. Sayed, S.A.F.; Nabil, E.; Badr, A. A binary clonal flower pollination algorithm for feature selection. *Pattern Recognit. Lett.* **2016**, *77*, 21–27. [CrossRef]
4. Martin-Bautista, M.J.; Vila, M.A. A survey of genetic feature selection in mining issues. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999; Volume 2, pp. 1314–1321.
5. Piramuthu, S. Evaluating feature selection methods for learning in data mining applications. *Eur. J. Oper. Res.* **2004**, *156*, 483–494. [CrossRef]
6. Gunal, S.; Edizkan, R. Subspace based feature selection for pattern recognition. *Inf. Sci.* **2008**, *178*, 3716–3726. [CrossRef]
7. Lew, M.S. *Principles of Visual Information Retrieval*; Springer Science & Business Media: Berlin, Germany, 2013.
8. Wang, G.G.; Tan, Y. Improving metaheuristic algorithms with information feedback models. *IEEE Trans. Cybern.* **2017**, *49*, 542–555. [CrossRef] [PubMed]
9. Houssein, E.H.; Hosney, M.E.; Elhoseny, M.; Oliva, D.; Mohamed, W.M.; Hassaballah, M. Hybrid Harris hawks optimization with cuckoo search for drug design and discovery in chemoinformatics. *Sci. Rep.* **2020**, *10*, 1–22. [CrossRef] [PubMed]
10. Houssein, E.H.; Hosney, M.E.; Oliva, D.; Mohamed, W.M.; Hassaballah, M. A novel hybrid Harris hawks optimization and support vector machines for drug design and discovery. *Comput. Chem. Eng.* **2020**, *133*, 106656. [CrossRef]

11. Gao, D.; Wang, G.G.; Pedrycz, W. Solving fuzzy job-shop scheduling problem using de algorithm improved by a selection mechanism. *IEEE Trans. Fuzzy Syst.* **2020**. [CrossRef]
12. Houssein, E.H.; Saad, M.R.; Hussain, K.; Zhu, W.; Shaban, H.; Hassaballah, M. Optimal sink node placement in large scale wireless sensor networks based on Harris' hawk optimization algorithm. *IEEE Access* **2020**, *8*, 19381–19397. [CrossRef]
13. Ahmed, M.M.; Houssein, E.H.; Hassanien, A.E.; Taha, A.; Hassanien, E. Maximizing lifetime of large-scale wireless sensor networks using multi-objective whale optimization algorithm. *Telecommun. Syst.* **2019**, *72*, 243–259. [CrossRef]
14. Chandrashekar, G.; Sahin, F. A survey on feature selection methods. *Comput. Electr. Eng.* **2014**, *40*, 16–28. [CrossRef]
15. Liu, H.; Yu, L. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. Knowl. Data Eng.* **2005**, *17*, 491–502.
16. Kohavi, R.; John, G.H. Wrappers for feature subset selection. *Artif. Intell.* **1997**, *97*, 273–324. [CrossRef]
17. Liu, H.; Setiono, R. A probabilistic approach to feature selection-a filter solution. In Proceedings of the 13th International Conference on Machine Learning, Bari, Italy, 3–6 July 1996; Volume 23, pp. 319–327.
18. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction;* Springer: New York, NY, USA, 2002.
19. Safavian, S.R.; Landgrebe, D. hastie2002elements. *IEEE Trans. Syst. Man, Cybern.* **1991**, *21*, 660–674. [CrossRef]
20. Dasarathy, B.V. *Nearest Neighbor ({NN}) Norms:{NN} Pattern Classification Techniques*; IEEE Computer Society Press: Washington, DC, USA, 1991.
21. Verikas, A.; Bacauskiene, M. Feature selection with neural networks. *Pattern Recognit. Lett.* **2002**, *23*, 1323–1335. [CrossRef]
22. Vapnik, V.N.; Vapnik, V. *Statistical Learning Theory*; Wiley: New York, NY, USA, 1998; Volume 1.
23. Khalid, S.; Khalil, T.; Nasreen, S. A survey of feature selection and feature extraction techniques in machine learning. In Proceedings of the Science and Information Conference (SAI), London, UK, 27–29 August 2014; pp. 372–378.
24. Wang, G.G.; Guo, L.; Gandomi, A.H.; Hao, G.S.; Wang, H. Chaotic krill herd algorithm. *Inf. Sci.* **2014**, *274*, 17–34. [CrossRef]
25. Hassanien, A.E.; Emary, E. *Swarm Intelligence: Principles, Advances, and Applications*; CRC Press: Boca Raton, FL, USA, 2016.
26. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
27. Hussien, A.G.; Amin, M.; Abd El Aziz, M. A comprehensive review of moth-flame optimisation: Variants, hybrids, and applications. *J. Exp. Theor. Artif. Intell.* **2020**, *32*, 705–725. [CrossRef]
28. Assiri, A.S.; Hussien, A.G.; Amin, M. Ant Lion Optimization: Variants, hybrids, and applications. *IEEE Access* **2020**, *8*, 77746–77764. [CrossRef]
29. Hussien, A.G.; Amin, M.; Wang, M.; Liang, G.; Alsanad, A.; Gumaei, A.; Chen, H. Crow Search Algorithm: Theory, Recent Advances, and Applications. *IEEE Access* **2020**, *8*, 173548–173565. [CrossRef]
30. Shareef, H.; Ibrahim, A.A.; Mutlag, A.H. Lightning search algorithm. *Appl. Soft Comput.* **2015**, *36*, 315–333. [CrossRef]
31. Hashim, F.A.; Houssein, E.H.; Mabrouk, M.S.; Al-Atabany, W.; Mirjalili, S. Henry gas solubility optimization: A novel physics-based algorithm. *Future Gener. Comput. Syst.* **2019**, *101*, 646–667. [CrossRef]
32. Houssein, E.H.; Saad, M.R.; Hashim, F.A.; Shaban, H.; Hassaballah, M. Lévy flight distribution: A new metaheuristic algorithm for solving engineering optimization problems. *Eng. Appl. Artif. Intell.* **2020**, *94*, 103731. [CrossRef]
33. Hashim, F.A.; Houssein, E.H.; Hussain, K.; Mabrouk, M.S.; Al-Atabany, W. A modified Henry gas solubility optimization for solving motif discovery problem. *Neural Comput. Appl.* **2020**, *32*, 10759–10771. [CrossRef]
34. Fernandes, C.; Pontes, A.; Viana, J.; Gaspar-Cunha, A. Using multiobjective evolutionary algorithms in the optimization of operating conditions of polymer injection molding. *Polym. Eng. Sci.* **2010**, *50*, 1667–1678. [CrossRef]
35. Gaspar-Cunha, A.; Covas, J.A. RPSGAe—Reduced Pareto set genetic algorithm: Application to polymer extrusion. In *Metaheuristics for Multiobjective Optimisation*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 221–249.

36. Avalos, O.; Cuevas, E.; Gálvez, J.; Houssein, E.H.; Hussain, K. Comparison of Circular Symmetric Low-Pass Digital IIR Filter Design Using Evolutionary Computation Techniques. *Mathematics* **2020**, *8*, 1226. [CrossRef]
37. Xue, B.; Zhang, M.; Browne, W.N.; Yao, X. A survey on evolutionary computation approaches to feature selection. *IEEE Trans. Evol. Comput.* **2016**, *20*, 606–626. [CrossRef]
38. Kabir, M.M.; Shahjahan, M.; Murase, K. A new hybrid ant colony optimization algorithm for feature selection. *Expert Syst. Appl.* **2012**, *39*, 3747–3763. [CrossRef]
39. Ghaemi, M.; Feizi-Derakhshi, M.R. Feature selection using forest optimization algorithm. *Pattern Recognit.* **2016**, *60*, 121–129. [CrossRef]
40. Emary, E.; Zawbaa, H.M.; Ghany, K.K.A.; Hassanien, A.E.; Parv, B. Firefly optimization algorithm for feature selection. In Proceedings of the 7th Balkan Conference on Informatics Conference, Craiova, Romania, 2–4 September 2015; p. 26.
41. Mafarja, M.M.; Mirjalili, S. Hybrid Whale Optimization Algorithm with simulated annealing for feature selection. *Neurocomputing* **2017**, *260*, 302–312. [CrossRef]
42. Xue, B.; Zhang, M.; Browne, W.N. Particle swarm optimization for feature selection in classification: A multi-objective approach. *IEEE Trans. Cybern.* **2013**, *43*, 1656–1671. [CrossRef]
43. Hafez, A.I.; Zawbaa, H.M.; Emary, E.; Hassanien, A.E. Sine cosine optimization algorithm for feature selection. In Proceedings of the 2016 International Symposium on INnovations in Intelligent Systems and Applications (INISTA), Sinaia, Romania, 2–5 August 2016; pp. 1–5.
44. Wang, G.G.; Deb, S.; Cui, Z. Monarch butterfly optimization. *Neural Comput. Appl.* **2019**, *31*, 1995–2014. [CrossRef]
45. Wang, G.G. Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Memetic Comput.* **2018**, *10*, 151–164. [CrossRef]
46. Rodrigues, D.; Yang, X.S.; De Souza, A.N.; Papa, J.P. Binary flower pollination algorithm and its application to feature selection. In *Recent Advances in Swarm Intelligence and Evolutionary Computation*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 85–100.
47. Nakamura, R.Y.; Pereira, L.A.; Costa, K.; Rodrigues, D.; Papa, J.P.; Yang, X.S. BBA: A binary bat algorithm for feature selection. In Proceedings of the 2012 25th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), Ouro Preto, Brazil, 22–25 August 2012; pp. 291–297.
48. Rodrigues, D.; Pereira, L.A.; Almeida, T.; Papa, J.P.; Souza, A.; Ramos, C.C.; Yang, X.S. BCS: A binary cuckoo search algorithm for feature selection. In Proceedings of the 2013 IEEE International Symposium on Circuits and Systems (ISCAS), Beijing, China, 19–23 May 2013; pp. 465–468.
49. He, X.; Zhang, Q.; Sun, N.; Dong, Y. Feature selection with discrete binary differential evolution. In Proceedings of the 2009 International Conference on Artificial Intelligence and Computational Intelligence, Shanghai, China, 7–8 November 2009; Volume 4, pp. 327–330.
50. Emary, E.; Zawbaa, H.M.; Hassanien, A.E. Binary ant lion approaches for feature selection. *Neurocomputing* **2016**, *213*, 54–65. [CrossRef]
51. Emary, E.; Zawbaa, H.M.; Hassanien, A.E. Binary grey wolf optimization approaches for feature selection. *Neurocomputing* **2016**, *172*, 371–381. [CrossRef]
52. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. BGSA: Binary gravitational search algorithm. *Nat. Comput.* **2010**, *9*, 727–745. [CrossRef]
53. Hussien, A.G.; Hassanien, A.E.; Houssein, E.H. Swarming behaviour of salps algorithm for predicting chemical compound activities. In Proceedings of the 2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, 5–7 December 2017; pp. 315–320.
54. Nezamabadi-pour, H.; Rostami-Shahrbabaki, M.; Maghfoori-Farsangi, M. Binary particle swarm optimization: Challenges and new solutions. *CSI J. Comput. Sci. Eng.* **2008**, *6*, 21–32.
55. Hussien, A.G.; Houssein, E.H.; Hassanien, A.E. A binary whale optimization algorithm with hyperbolic tangent fitness function for feature selection. In Proceedings of the 2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, 5–7 December 2017; pp. 166–172.
56. Hussien, A.G.; Hassanien, A.E.; Houssein, E.H.; Bhattacharyya, S.; Amin, M. S-shaped Binary Whale Optimization Algorithm for Feature Selection. In *Recent Trends in Signal and Image Processing*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 79–87.
57. Hussien, A.G.; Hassanien, A.E.; Houssein, E.H.; Amin, M.; Azar, A.T. New binary whale optimization algorithm for discrete optimization problems. *Eng. Optim.* **2020**, *52*, 945–959. [CrossRef]

58. Chuang, L.Y.; Chang, H.W.; Tu, C.J.; Yang, C.H. Improved binary PSO for feature selection using gene expression data. *Comput. Biol. Chem.* **2008**, *32*, 29–38. [CrossRef]
59. Dua, D.; Graff, C. UCI Machine Learning Repository. 2017. Available online: http://archive.ics.uci.edu/ml (accessed on 28 September 2020).
60. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43.
61. Mafarja, M.M.; Eleyan, D.; Jaber, I.; Hammouri, A.; Mirjalili, S. Binary dragonfly algorithm for feature selection. In Proceedings of the 2017 International Conference on New Trends in Computing Sciences (ICTCS), Amman, Jordan, 11–13 October 2017; pp. 12–17.
62. D'Angelo, G.; Palmieri, F. GGA: A modified Genetic Algorithm with Gradient-based Local Search for Solving Constrained Optimization Problems. *Inf. Sci.* **2020**, *547*, 136–162. [CrossRef]
63. Mirjalili, S.; Hashim, S.Z.M. BMOA: Binary magnetic optimization algorithm. *Int. J. Mach. Learn. Comput.* **2012**, *2*, 204. [CrossRef]
64. Alon, U.; Barkai, N.; Notterman, D.A.; Gish, K.; Ybarra, S.; Mack, D.; Levine, A.J. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. USA* **1999**, *96*, 6745–6750. [CrossRef] [PubMed]
65. Alizadeh, A.A.; Eisen, M.B.; Davis, R.E.; Ma, C.; Lossos, I.S.; Rosenwald, A.; Boldrick, J.C.; Sabet, H.; Tran, T.; Yu, X.; et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature* **2000**, *403*, 503–511. [CrossRef] [PubMed]
66. Golub, T.R.; Slonim, D.K.; Tamayo, P.; Huard, C.; Gaasenbeek, M.; Mesirov, J.P.; Coller, H.; Loh, M.L.; Downing, J.R.; Caligiuri, M.A.; et al. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science* **1999**, *286*, 531–537. [CrossRef] [PubMed]
67. Wilcoxon, F. Individual comparisons by ranking methods. *Biom. Bull.* **1945**, *1*, 80–83. [CrossRef]

mathematics

MDPI

Article

A Memetic Decomposition-Based Multi-Objective Evolutionary Algorithm Applied to a Constrained Menu Planning Problem

Alejandro Marrero [1,*], Eduardo Segredo [1], Coromoto León [1] and Carlos Segura [2]

[1] Departamento de Ingeniería Informática y de Sistemas, Universidad de La Laguna, Apto. 456. 38200 San Cristóbal de La Laguna, Tenerife, Spain; esegredo@ull.edu.es (E.S.); cleon@ull.edu.es (C.L.)
[2] Área de Computación, Centro de Investigación en Matemáticas, A.C. 36023 Guanajato, Mexico; carlos.segura@cimat.mx
* Correspondence: amarrerd@ull.edu.es

Received: 1 October 2020; Accepted: 2 November 2020; Published: 5 November 2020

Abstract: Encouraging healthy and balanced diet plans is one of the most important action points for governments around the world. Generating healthy, balanced and inexpensive menu plans that fulfil all the recommendations given by nutritionists is a complex and time-consuming task; because of this, computer science has an important role in this area. This paper deals with a novel constrained multi-objective formulation of the menu planning problem specially designed for school canteens that considers the minimisation of the cost and the minimisation of the level of repetition of the specific courses and food groups contained in the plans. Particularly, this paper proposes a multi-objective memetic approach based on the well-known multi-objective evolutionary algorithm based on decomposition (MOEA/D). A crossover operator specifically designed for this problem is included in the approach. Moreover, an ad-hoc iterated local search (ILS) is considered for the improvement phase. As a result, our proposal is referred to as ILS-MOEA/D. A wide experimental comparison against a recently proposed single-objective memetic scheme, which includes explicit mechanisms to promote diversity in the decision variable space, is provided. The experimental assessment shows that, even though the single-objective approach yields menu plans with lower costs, our multi-objective proposal offers menu plans with a significantly lower level of repetition of courses and food groups, with only a minor increase in cost. Furthermore, our studies demonstrate that the application of multi-objective optimisers can be used to implicitly promote diversity not only in the objective function space, but also in the decision variable space. Consequently, in contrast to the single-objective optimiser, there was no need to include an explicit strategy to manage the diversity in the decision space in the case of the multi-objective approach.

Keywords: menu planning problem; evolutionary algorithm; decomposition-based multi-objective optimisation; memetic algorithm; iterated local search; diversity preservation

1. Introduction

Nowadays, due to unhealthy and sedentary lifestyles, a high percentage of the human population suffers from various diseases such as high cholesterol, diabetes and other conditions related to those habits which can cause other serious illnesses, including several types of cancer [1]. As a result, promoting healthy and active habits for young people is becoming a significant duty for governments around the world. Since it is impossible for governments to control what children and teenagers consume in their homes, having a healthy and balanced meal plan for school cafeterias is essential to help mitigate the effects of the remaining meals. The Government of the Canary Islands wants to promote healthy habits in children and teenagers. Generating healthy and balanced menu plans is

key to this effort. This is precisely the main goal of this research, which is part of the "Programa de Eco-comedores Escolares de Canarias" programme, which seeks to generate healthy, balanced and affordable menu plans for regional school cafeterias.

This work presents a novel constrained multi-objective formulation of the well-known menu planning problem (MPP) [2]. Specifically, the version of the MPP considered herein is a multi-objective formulation that includes a set of daily and n-days constraints for several nutrients, as introduced in [3], as well as the two objective functions related to the cost of the menu and the level of repetition of courses and food groups proposed in [4].

Planning and scheduling problems have been successfully solved with meta-heuristics. Particularly, many of the best-known solutions for problems in this area have been achieved with memetic algorithms (MAs) [5–8]. In [3], an MA was proposed to deal with a single-objective constrained formulation of the MPP where the cost of the menu had to be minimised. To do so, a specific iterated local search (ILS) was designed, as well as an ad-hoc crossover operator. Additionally, the MA included an explicit mechanism to promote diversity in the decision variable space in order to avoid premature convergence. The above allowed high-quality solutions to be attained. The working hypothesis herein is that by using our novel multi-objective constrained formulation of the MPP, which considers the cost of the plan and at the same time the level of repetition of the specific courses and food groups contained in the plan, it is possible to find solutions that are similar in terms of the cost to those provided by the aforementioned single-objective MA, but significantly better regarding the level of repetition.

Taking these findings into account, this work presents a novel multi-objective approach based on the well-known multi-objective evolutionary algorithm based on decomposition (MOEA/D) [9]. MOEA/D was applied to the menu planning problem in a previous work carried out by the authors [10]. It was compared to the well-known Non-dominated Sorting Genetic Algorithm II (NSGA-II) [11] and Strength Pareto Evolutionary Algorithm 2 (SPEA2) [12]. Since knowledge about the menu planning problem was not considered, ad-hoc variation operators, as well as an improvement operator, were not incorporated into MOEA/D. Results showed that MOEA/D was outperformed by NSGA-II and SPEA2. In order to improve the performance of MOEA/D when dealing with the menu planning problem, in the current work, an extension of the ILS applied by the single-objective MA proposed in [3] was considered as the improvement operator of MOEA/D. We should note that, in opposition to other approaches, MOEA/D facilitates the incorporation of an improvement operator, which is an important component to quickly yield feasible solutions. Finally, the ad-hoc crossover operator proposed by the same authors was also integrated into MOEA/D.

As a result, our proposal is referred to as ILS-MOEA/D, and in contrast to the single-objective optimiser presented in [3], it does not include an explicit strategy to manage diversity in the decision variable space. The reason is that, as in other problems [13], the promotion of diversity in the objective function space, imposed by the use of different weights in ILS-MOEA/D to decompose the problem, causes an implicit preservation of diversity in the decision variable space. A wide experimental evaluation is carried out herein, where the novel ILS-MOEA/D is compared against the single-objective MA proposed in [3], in terms of the quality of the solutions attained by both optimisers.

Bearing all the above in mind, the main contributions of this work are the following:

- A novel constrained multi-objective formulation of the MPP.
- A novel adaptation of MOEA/D, ILS-MOEA/D, that integrates an ILS as the improvement step of the approach, as well as an ad-hoc crossover operator to speed up the achievement of high-quality solutions for the multi-objective constrained version of the MPP.
- In comparison to the single-objective MA, ILS-MOEA/D yields menu plans with a similar cost, but with a significantly better level of repetition.
- The promotion of diversity in the objective function space through the application of a multi-objective algorithm helps avoid premature convergence, since diversity is also implicitly kept in the decision variable space.

The rest of this paper is structured as follows. Section 2 provides a review of related works and presents our new formulation of the MPP. Afterwards, the novel ILS-MOEA/D is detailed in Section 3. Then, the experimental evaluation performed to validate our proposals is presented in Section 4. The results obtained from the experiments are discussed in Section 5. Finally, the conclusions are exposed in Section 6, together with some lines of further research.

2. Menu Planning Problem

2.1. Background

Menu planing has been solved by using computers since early 1960 [2,14]. Many of the proposed formulations are NP-complete, meaning it is quite a complex task [15]. In essence, the classical MPP aims to find a combination of dishes which satisfies certain restrictions involving budget, variety and nutritional requirements for an n-day sequence. Even though there is no consensus on the specific objectives that an MPP formulation should consider, in almost every formulation, it is common for the cost of the menu plan to be considered one of the main objectives to optimise [2–4,10,16,17]. Other options for defining the objective functions are the users' preferences for certain foods, the level of adequacy or the level of acceptance [18–21].

The constraints considered in menu planning problems are usually based on the nutritional requirements that meal plans have to satisfy. As a result a set of constraints is defined that models the recommended minimum and maximum amounts of different nutrients [21–26]. Other constraints consider the variety of the meals, their predominant colour, their consistency, the time required to prepare them and food that cannot be consumed, among others [19,22,25,27]. Two of the most frequently used techniques to handle constraints are based on the application of repairing methods [4,10,19,26,28,29] or penalisation functions [16,30,31]. In the first case, operators are applied to an infeasible solution until it becomes feasible. In the second case, the fitness function is penalised somehow depending on the degree of infeasibility of the corresponding solution. Hence, the higher the degree of infeasibility of the solution, the larger the probability to be discarded.

Both single-objective [3,16,19,32] and multi-objective optimisers [4,10,17,28] have been devised for the MPP. In the single-objective case, most of the formulations consider the cost as the only objective to optimise, while the nutritional requirements are used as constraints. In the case of multi-objective formulations, the cost is always considered as one of the objective functions to optimise [29–31]. Additionally, the seasonal quality, food flavour and food temperature [29], food preferences [30] and the nutritional error [31] are considered as other objective functions. In almost all cases, the nutritional requirements, as well as the users' personal preferences, model the constraints of the multi-objective formulations.

Although there exist many different types of algorithms for solving this problem, a high percentage of published papers use evolutionary algorithms (EAs) or other types of meta-heuristics due to the benefits they offer, such as robustness, reliability, global-search ability and simplicity [16,20,21,26,28]. EAs are approximated methods based on the concept in natural evolution of survival of the fittest individual [33]. Given a population of individuals in some environment with limited resources, the competition for survival causes natural selection, with the fittest individuals more likely to survive and reproduce. In addition, being approximated methods means that although there is no guarantee of obtaining the optimal solution to a problem, high-quality solutions can be found in a reasonable period of time. The classical genetic algorithm (GA) is the most common approach for solving single-objective formulations of the MPP [16,34], while previous multi-objective formulations of the MPP have been frequently addressed by applying the state-of-art NSGA-II, such as the work proposed in [28,30,31].

The lack of ad-hoc operators for the MPP could possibly yield sub-optimal solutions due to the problem of premature convergence. In order to avoid this, problem-specific operators or procedures, such as intensification mechanisms, have been included into EAs, resulting in memetic algorithms (MAs) [35,36]. MAs can be seen as the combination of an EA with an intensification mechanism in

order to improve the general performance of the optimiser [37]. An example of a single-objective MA successfully applied to the MPP is that proposed in [3].

Additionally, a software named SCHOOLTHY was proposed in [4], which allows menus to be planned automatically. The tool uses an MPP formulation similar to the one proposed in this work to generate not only affordable plans, but also varied from a nutritional standpoint. Although it was designed for school cafeterias, it could be adapted to other environments, such as hospitals, prisons and retirement homes, among others.

As we previously mentioned in Section 1, in the current work, we present a novel constrained multi-objective formulation of the MPP. It consists of the same set of nutritional daily and n-days (global) constraints presented in [3]. At the same time, the two objective functions proposed in [4], i.e., the cost of the meal plan and its level of repetition of courses and food groups, are also considered. As far as we know, this is the first time that an objective function modelling the level of repetition of specific courses and food groups is optimised together with the cost under a multi-objective formulation of the menu planning problem, which in addition considers the management of daily and global nutritional constraints. Those constraints are successfully managed by considering the infeasibility degree of a particular solution, thus giving preference to those solutions with, first, a lower infeasibility degree, and second, a lower fitness in terms of both aforementioned objective functions. Furthermore, the application of a multi-objective memetic approach based on decomposition that includes knowledge about the menu planning problem in the form of a tailored improvement operator and an ad-hoc crossover operator had never been carried out before. The above allows feasible solutions of the multi-objective constrained formulation that we are presenting herein to be attained quickly.

2.2. Formulation

In this work, a novel multi-objective constrained formulation of the MPP focused on school cafeterias is proposed. Due to the above, only lunch is planned for n days, including a starter, a main course and a dessert per day. The different courses are selected from a database containing their cost and nutritional facts. Courses in the database are grouped into three different categories: starters, main courses and desserts. The particular objective functions are, on the one hand, the cost of the whole plan, which has to be minimised, and on the other hand, the level of repetition of specific courses and food groups that the plan consists of, which has to be minimised as well. The motivation behind the second objective function is to promote a varied plan from a nutritional point of view.

Additionally, the set of daily and global constraints applied in [3] are also taken into account in this new formulation. Those constraints are modelled on the recommendations on intakes of macro-nutrients and micro-nutrients following the guidelines given in the White Book on Child Nutrition. Furthermore, other reference documents regarding school diets and allergens were also consulted, specifically, those endorsed by the Spanish Government Ministry of Education, Culture and Sports and the Ministry of Health, Social Services and Equality (These consensus documents are available at http://www.aecosan.msssi.gob.es/AECOSAN/docs/documentos/nutricion/educanaos/documento_consenso.pdf and https://www.aepnaa.org/).

As we said, the cost is an objective to be minimised and is calculated as the sum of the costs of the courses included in the plan. Note that the cost of each course is calculated as the sum of the costs of all the ingredients required to prepare that course. Formally, the cost is defined as follows:

$$C = \sum_{j=1}^{n} \left(c_{fc_j} + c_{sc_j} + c_{ds_j} \right) \tag{1}$$

where C is the total cost of the menu plan and $c_{fc_j}, c_{sc_j}, c_{ds_j}$ represent the cost of the starter, main course and dessert, respectively, for day j, and n is the number of days for which the menu plan is being designed.

An assorted menu plan is particularly important when it is intended for children. As a result of this, the level of repetition of courses and food groups was defined as the second objective function to

be minimised. The level of repetition represents the percentage of courses and food groups repeated throughout the meal plan. The following equation defines how it is calculated:

$$L_{Rep} = \sum_{j=1}^{n} \left(v_{MC_j} + \frac{p_{fc}}{d_{fc_j}} + \frac{p_{sc}}{d_{sc_j}} + \frac{p_{ds}}{d_{ds_j}} + v_{FG_j} \right) \tag{2}$$

where L_{Rep} is the level of repetition, v_{MC_j} represents the compatibility, in terms of food groups, among courses fc_j, sc_j, ds_j for day j; p is a penalty constant for every kind of course and d stands for the number of days since a specific course was repeated. Finally, v_{FG_j} is the penalty value for repeating particular food groups in the last five days.

Equation (3) allows the value of v_{MC_j} to be calculated, where $|G|$ is the number of food groups, $x_g \in \{0, 1, 2, 3\}$ is the number of times a particular food group is contained in the three courses (starter, main course and dessert) of the menu for day j and p_g is the corresponding penalty value for repeating the food group g. The food groups considered in this work are as follows: meat, cereal, fruit, dairy, legume, shellfish, pasta, fish, vegetable and other.

$$v_{MC_j} = \sum_{g=1}^{|G|} \left(x_{g_j} \cdot p_g \right) \tag{3}$$

$$v_{FG_j} = \sum_{i=1}^{min(j-1,T)} \left((\sum_{g=1}^{|G|} x_{g_i} \cdot p_g) + (y_i \cdot p_{|G|+i}) \right) \tag{4}$$

Equation (4) is used to compute v_{FG_j}, where $T = 5$ days is the number of previous days considered, $|G|$ is the number of food groups, $x_{g_i} \in \{0, 1\}$ indicates whether the food group g is repeated on day $j - i$ ($x_{g_i} = 1$) or not ($x_{g_i} = 0$) with respect to day j, $y_i \in \{0, 1\}$ indicates whether any food group was repeated i day(s) before the current day j ($y_i = 1$) or not ($y_i = 0$), and p_g and $p_{|G|+i}$ are the corresponding penalty values.

The types of penalties and their values used to compute v_{MC_j} and v_{FG_j} are shown in Table 1. Furthermore, penalties are determined by the repetition of food groups (p_1–p_{10}), the repetition of the same food group from one to five days prior to the current day (p_{11}–p_{15}), and the repetition of specific courses ($p_{16} = p_{fc}$, $p_{17} = p_{sc}$, and $p_{18} = p_{ds}$). In the case of penalties for repeating food groups (p_1–p_{10}), if the penalty value of a given food group is very large in comparison to the remaining food group penalty values, then a plan with a lower number of courses belonging to that food group will be provided. For instance, we have given preference to those courses consisting primarily of vegetables ($p_{10} = 0.1$) over courses composed primarily of meat ($p_2 = 3$).

Additionally, as stated earlier, in order to consider a menu plan as feasible, it must fulfil some constraints related to a set of nutritional requirements, such as having each nutrient intake be within a given range. Besides, the set of constraints of this MPP formulation is modelled by two sub-sets of constraints: global constraints and daily constraints. For instance, energy (kcal), fats and proteins are evaluated both daily and globally. At this point, we note that, since only lunch is considered in the meal plans, the recommended nutritional intakes were adapted. For each nutrient h considered, r_h denotes the recommended amount to ingest every day at lunch. Based on the recommended amount, a range of acceptable intake is generated for each nutrient. Table 2 defines a set R of pairs (r_{min}, r_{max}) with the minimum and maximum amount allowed for each nutrient h, respectively (The set of micro-nutrients is as follows: Folic acid, Phosphorus, Magnesium, Selenium, Sodium, Vitamins A, B1, B2, B6, B12, C, D, E, Iodine, Zinc).

Table 1. Types of penalties defined to calculate the level of repetition objective.

Penalty	Description	Value
p_1	penalty for repeating other food group	0.1
p_2	penalty for repeating meat food group	3
p_3	penalty for repeating cereal food group	0.3
p_4	penalty for repeating fruit food group	0.1
p_5	penalty for repeating dairy food group	0.3
p_6	penalty for repeating legume food group	0.3
p_7	penalty for repeating shellfish food group	2
p_8	penalty for repeating pasta food group	1.5
p_9	penalty for repeating fish food group	0.5
p_{10}	penalty for repeating vegetable food group	0.1
p_{11}	penalty for repeating the same food group one day earlier	3
p_{12}	penalty for repeating the same food group two days earlier	2.5
p_{13}	penalty for repeating the same food group three days earlier	1.8
p_{14}	penalty for repeating the same food group four days earlier	1
p_{15}	penalty for repeating the same food group five days earlier	0.2
$p_{16} = p_{fc}$	penalty for repeating a starter	8
$p_{17} = p_{sc}$	penalty for repeating a main course	10
$p_{18} = p_{ds}$	penalty for repeating a dessert	2

Table 2. Minimum and maximum ranges for nutrient intakes

Nutrient	r_{min}	r_{max}
Energy (per day)	0.85	1.15
Fats (per day)	0.75	1.25
Proteins (per day)	0.75	1.25
Energy (n days)	0.90	1.10
Fats (n days)	0.90	1.10
Proteins (n days)	0.90	1.10
Micro-nutrients	0.70	1.30

Formally, an individual S would be considered feasible if and only if it satisfies the following set of global constraints:

$$\forall h \in HG : n \times r_{\min_h} \times r_h \leq in(S,h) \leq n \times r_{max_h} \times r_h \tag{5}$$

where $in(S,h)$ is the global intake of nutrient h in the plan S, and HG denotes the set of nutrients considered for the global constraints.

In the case of energy, fats and proteins, their intakes for every single day d, i.e., $in(S,h,d)$ are also checked to be in the established daily ranges:

$$\forall h \in HD : r_{\min_h} \times r_h \leq in(S,h,d) \leq r_{max_h} \times r_h \tag{6}$$

where HD is the set of nutrients considered for the daily constraints.

In order to properly compare solutions, a definition of an infeasibility degree id is required. As previously defined in [3], the infeasibility degree of a solution S is calculated as shown in Equation (7). Note that an individual S that satisfies Equations (5) and (6) would have an infeasibility degree $id(S) = 0$, and it would be considered a feasible solution.

$$id(S) = gid(S) + did(S) \tag{7}$$

The infeasibility degree id is calculated as the sum of the global infeasibility degree $gid(S)$ and the daily infeasibility degree $did(S)$. Equations (8) and (9) show the calculation of $gid(S)$ and $did(S)$, respectively.

$$\begin{aligned} \text{gid}(S) = & \sum_{h \in HG} \left(\max\left(in(S,h), r_{max_h}\right) - r_{max_h}\right)^2 \times 10^6 + \\ & \sum_{h \in HG} \left(r_{min_h} - \min\left(in(S,h), r_{min_h}\right)\right)^2 \times 10^6 \end{aligned} \tag{8}$$

$$\begin{aligned} \text{did}(S) = & \sum_{h \in HD} \sum_{d=1}^{n} \left(\max\left(in(S,h,d), r_{max_h}\right) - r_{max_h}\right)^2 + \\ & \sum_{h \in HD} \sum_{d=1}^{n} \left(r_{min_h} - \min\left(in(S,h,d), r_{min_h}\right)\right)^2 \end{aligned} \tag{9}$$

3. Algorithms

The method that we are presenting herein (ILS-MOEA/D) is a particularisation of MOEA/D that has been specifically designed to deal with the multi-objective constrained MPP proposed in the previous section. MOEA/D is an EA for multi-objective optimisation proposed in [9], where the underlying idea is to decompose a Multi-objective Optimisation Problem (MOP) into a number of scalar optimisation sub-problems and optimise them simultaneously.

MOEA/D is a population-based EA that maintains N candidate solutions in the population. The decomposition approach is applied to generate N sub-problems that are simultaneously optimised. Specifically, each individual is associated to one of the generated sub-problems (a one-to-one mapping is performed). Furthermore, it establishes some relationships between sub-problems and organises them into neighbourhoods. These neighbourhoods are defined in terms of the weight vectors used to decompose the problem. The optimisation of each sub-problem is influenced by information of its neighbouring sub-problems. The principle for organising the optimisation process in this way is that the optimal solution for two neighbouring sub-problems is expected to be similar [9]. Furthermore, the process of decomposing an MOP into N scalar optimisation sub-problems can be done by applying different approaches.

In this paper, our strategy relies on the Tchebycheff decomposition approach [38]. Let $\lambda = \{\lambda^1, \ldots, \lambda^N\}$ be a set of even spread weight vectors. In this approach, the scalar objective function associated to the j-th sub-problem, with $j = 1, \ldots, N$, is defined as follows:

$$g^{te}(x|\lambda^j, z^*) = max_{i=1}^{m}\{\lambda_i^j | f_i(x) - z_i^*|\} \tag{10}$$

where $x \in \Omega$ is a solution to a multi-objective problem with m original objectives; $z^* = (z_1^*, \ldots, z_m^*)$ is the reference point with the best solution found so far for each of the $i = 1, \ldots, m$ original objectives and $\lambda^j = (\lambda_1^j, \ldots, \lambda_m^j)$. We note that MOEA/D minimises all those N scalar optimisation sub-problems simultaneously.

Since the MPP formulation considered in this work implies handling the feasibility constraints, the replacement of any individual from the population must satisfy certain criteria. For example, given two individuals S_1 and S_2:

- If their infeasibility degrees are different, the one with a lower infeasibility degree is better.
- Otherwise, if two individuals have the same infeasibility degree, i.e., $id(S_1) = id(S_2)$, the individual with better fitness is preferred. The fitness of an individual is computed using Equation (10).

Moreover, our novel ILS-MOEA/D includes two main improvements to speed up the achievement of high-quality solutions. First, the Similarity-based Crossover (SX) proposed in [3] is used as the only genetic operator that produces new individuals. Second, an adapted version of the ILS (Algorithm 1), which was also proposed in [3], was integrated as the intensification phase of ILS-MOEA/D. This ILS

is based on the well-known First-Improvement Hill Climbing approach (line 3). It explores the neighbourhood of a solution in a random way by accepting any movement that improves the current solution, and stopping after reaching a local optimum. The key difference of the ILS applied herein in comparison to the one proposed in [3] is that the procedure evaluates each improvement by using the Tchebycheff decomposition approach (Equation (10)) in a similar way as MOEA/D. Furthermore, the reference point z^* is also updated if any objective improves the current reference point In addition, for any solution, instead of using the neighbourhood structure of MOEA/D, the neighbours are generated by modifying a single course of the menu plan. As a result, the number of neighbours of any solution is $n \times (l_{fc} + l_{sc} + l_{ds} - 3)$, where l_{fc}, l_{sc} and l_{ds} are the number of options for starters, main courses and desserts, respectively. Finally, the procedure Perturb makes use of problem-dependent information of the MPP, since it identifies the potential problems in a menu plan in order to mitigate them. However, this procedure does not guarantee that the algorithm will not stagnate in local optima solutions. Additionally, since this MPP formulation has not been solved with any exact method, there is not any certainty that the solutions found by either ILS-MOEA/D or MA are global optima. The Perturb (line 7) procedure is the same as the one applied in [3]. The above steps are repeated *Iterations* times.

Algorithm 1: Iterated Local Search — ILS

```
   Input: S (Initial Solution), Iterations, λ^S, z*
1  Best = S;
2  for i ← 0 to Iterations do
3  |  S' = First-Improvement-Hill-Climbing(S, λ^S, z*);
4  |  if S' is Better than Best then
5  |  |  Best = S';
6  |  end
7  |  S = Perturb(Best);
8  end
9  return Best
```

The proposed ILS-MOEA/D outlined in Algorithm 2 involves similar steps as the standard MOEA/D [9]:

1. The algorithm receives the population size N, the neighbourhood size L and a set of uniform sparse weight vectors λ, as the parameters to set up the run.
2. In line 1, an external population, EP, is created to store the non-dominated individuals found during the search. Since the population size parameter was set to a small value in this work, the external population is used to keep additional solutions to those maintained in the current population. An unbounded EP that maintains feasible non-dominated individuals is used.
3. Then, a set of neighbourhoods B is created in line 2. For each individual j in the population, its corresponding neighbourhood B_j is created, considering the L closest weight vectors to λ^j. The above is performed by computing the Euclidean distance between any two weight vectors.
4. After that, in line 3, the reference point z^* is initialised using the opposite largest value to the corresponding objective function direction, i.e., the maximum floating point number allowed for an objective function that has to be minimised and vice-versa.
5. The last step before starting the MOEA/D updating phase is to initialise the population (line 4). This procedure creates N new individuals randomly and then applies the aforementioned ILS to improve the quality of the initial solutions. Additionally, the reference point z is updated if required.

Algorithm 2: ILS-MOEA/D

```
   Input: N, L, λ
1  EP = ∅;
2  B = GenerateNeighbourhood(N, L, λ);
3  z* = InitialiseZ();
4  Population = InitialisePopulation(N, λ, z*);
5  while StoppingCriterion is not satisfied do
6      for j ← 1 to N do
7          Offspring = Reproduce(B, j);
8          Offspring = ILS(Offspring, 100, λ^j, z*);
9          z* = UpdateZ(Offspring);
10         Population = UpdateNeighbouringSolutions(Offspring);
11         EP = UpdateExternalPopulation(Offspring);
12     end
13 end
14 return EP
```

Afterwards, the main loop of the MOEA/D performs the updating phase of the algorithm by carrying out the following steps until the stopping criterion is met:

1. For each individual j in the population (line 6), a new offspring is created by applying the SX crossover operator to two neighbours of individual j (line 7), which are selected at random from its neighbourhood B_j. Afterwards, the ILS is applied to the new offspring (line 8).
2. Then, in line 9, the reference point z^* is updated if better objective values were found.
3. Next, in line 10, the offspring created is considered to replace individual j in the population by following the aforementioned replacement criteria. Note that each offspring at most replaces one individual in the population.
4. Finally, in line 11, the offspring is considered for insertion into the EP if its infeasibility degree is equal to zero and there is no individual in EP that dominates it.

4. Experimental Assessment

Due to the fact that the same constraints and courses database used in [3] is applied in this work, the results obtained by ILS-MOEA/D were compared, in terms of the meal plan cost, with those attained by the single-objective MA proposed in [3], which will be referred to as MA in the rest of the paper. Moreover, the level of repetition of the solutions provided by MA was also computed to properly evaluate the difference between the two approaches.

Regarding the courses database, a total of 64 different courses were available, grouped into three different categories: l_{fc}: 18 starters, l_{sc}: 33 main courses, l_{ds}: 13 desserts. Moreover, for every course available, the following information was obtained: name of the course, cost of the course, amount of nutrients in the course and the particular food groups the course belongs to.

Standard configurations for MA and ILS-MOEA/D were applied to different instances of the MPP. Specifically, plans for $n = 20$, $n = 40$ and $n = 60$ days were considered. All the experiments were performed using the elapsed time as the stopping criterion, which was different depending on the instance size. For $n = 20$ days, the stopping criterion was set to one hour, and it was increased to two and a half hours and five hours for $n = 40$ and $n = 60$ days, respectively.

Regarding the parameterisation of ILS-MOEA/D, the population size was set to $N = 15$ individuals, the neighbourhood size was fixed to $L = 5$ individuals and the crossover rate was set to $CR = 1.0$. The set of weight vectors λ was generated using the method described in [9]. We note that the same parameterisation was used to apply the single-objective MA, which means that the

population size was set to $N = 15$ individuals and the crossover rate $CR = 1.0$. None of the algorithms applied a mutation operator. Finally, the number of iterations of the ILS was set to 100 in both cases.

The Metaheuristic-based Extensible Tool for Cooperative Optimisation (METCO) (Available at: https://github.com/PAL-ULL/software-metco), described in [39], was used to implement ILS-MOEA/D and the multi-objective constrained formulation of the MPP discussed here, as well as to perform the entire experimental assessment.Experiments were run on a server belonging to the "Laboratorio de Supercómputo del Bajío", which is maintained by the "Centro de Investigación en Matemáticas" (CIMAT), Mexico. The server provides two Intel Xeon E5-2620 v2 processors with 6 cores each at 2.1 GHz with 32 GB RAM. Since we are dealing with stochastic approaches, every execution was repeated 30 times. In order to statistically support the conclusions, the following statistical testing procedure, which was formerly used in previous work by the authors [40], was applied to conduct comparisons between experiments. First, a Shapiro–Wilk test was performed to check whether the values of the results followed a normal (Gaussian) distribution. If so, the Levene test checked for the homogeneity of the variances. If the samples had equal variances, an ANOVA test was done; if not, a Welch test was performed. For non-Gaussian distributions, the non-parametric Kruskal–Wallis test was used. For every test, a significance level $\alpha = 0.05$ was considered.

The hypervolume (HV) [41], normalised in the range $[0, 1]$, was selected as the metric to measure the performance of ILS-MOEA/D. To compute the normalised HV, the resulting Pareto Fronts of ILS-MOEA/D were normalised using the worst and best values achieved for each objective function, which defines the lower and upper bounds among all the executions performed. We should note that since the HV metric has to be maximised, the higher its value, the better the performance.

5. Discussion

In this section, the results from the experimental assessment introduced previously are discussed (The results, plots and source code are available at https://github.com/Tomas-Morph/MenuPlanning_MOEAD_Mathematics). As mentioned earlier, the results attained by ILS-MOEA/D were compared to the results achieved by MA. The reader should recall that the working hypothesis behind this comparison is that by applying ILS-MOEA/D to a multi-objective formulation of the MPP, which considers not only the cost of the plan but also the level of repetition, it is possible to find solutions that are similar in terms of the cost to those provided by MA, but significantly better with respect to the level of repetition of specific courses and food groups contained in the menu plan.

First of all, MA seeks to obtain the cheapest menu plan that satisfies all the constraints defined in Section 2. Consequently, the menu plans generated by MA had a higher level of repetition in comparison to those provided by ILS-MOEA/D, as shown in Figure 1. The statistical procedure introduced previously shows that there were significant statistical differences in the results of ILS-MOEA/D and MA. Although an explicit diversity management strategy was implemented in MA, not considering the level of repetition in the single-objective formulation of the MPP led to more affordable but less varied menu plans. In fact, in terms of the cost, the results obtained by MA were statistically better than those achieved by ILS-MOEA/D for the three instances considered.

Since ILS-MOEA/D considers the level of repetition as one of the objective functions to be optimised, the menu plans generated by this algorithm had a better ratio between the cost and the level of repetition. Considering the level of repetition, ILS-MOEA/D provided statistically significant better results than MA for the three instances. Furthermore, as Figure 1 shows, although the results of ILS-MOEA/D in terms of the cost were noticeably more scattered than those attained by MA, in a few executions, ILS-MOEA/D yielded the best menu plan cost obtained by MA with a significantly lower level of repetition, in the case of $n = 20$ and $n = 40$ days. As the results in Table 3 show, for $n = 20$ days, the mean cost of the solutions provided by ILS-MOEA/D was only 0.17% higher in comparison to the mean cost of the solutions attained by MA. However, the mean level of repetition of the solutions obtained by MA was 28.3% higher when compared to the mean level of repetition of the solutions given by ILS-MOEA/D.

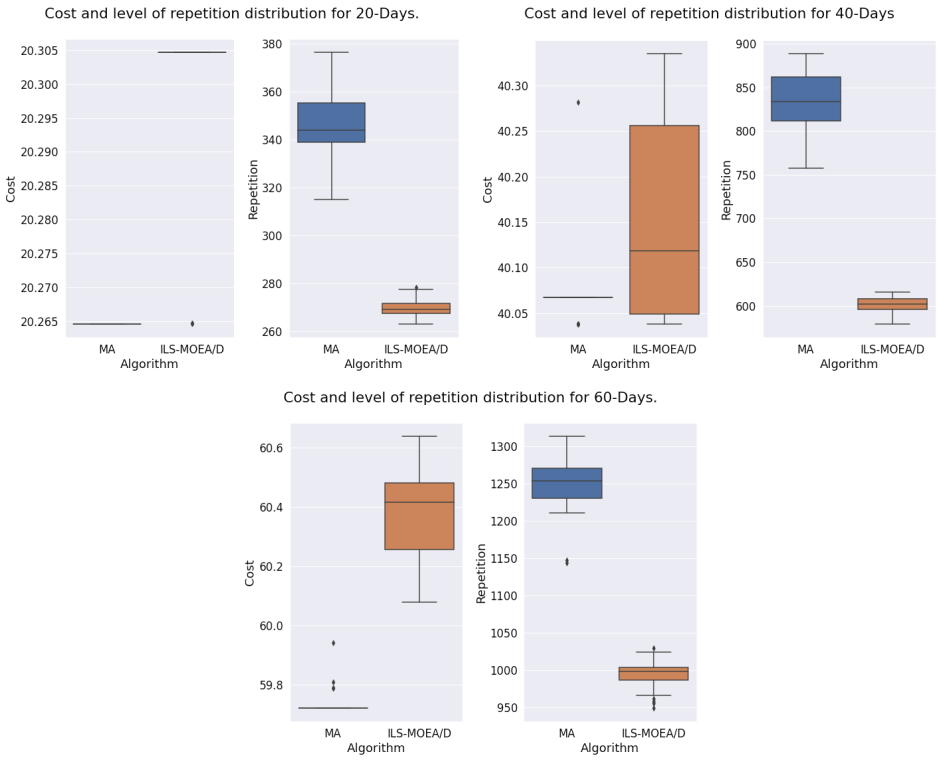


Figure 1. Boxplot representation of the values obtained by memetic algorithm (MA) and iterated local search (ILS)-multi-objective evolutionary algorithm based on decomposition (MOEA/D) for both objective functions, i.e., cost and level of repetition of specific courses and food groups, considering menu plans for $n = 20$ (**upper-left**), $n = 40$ (**upper-right**) and $n = 60$ (**bottom**) days.

Something similar happened for the menu plans for $n = 40$ days. In this case, the mean cost of the solutions provided by ILS-MOEA/D was only 0.21% higher in comparison to the mean cost of the solutions attained by MA. The mean level of repetition of the solutions obtained by MA, however, was 38.5% higher when compared to the mean level of repetition of the solutions given by ILS-MOEA/D. Finally, for $n = 60$ days, the mean cost provided by ILS-MOEA/D was only 1.05% higher in comparison to the mean cost attained by MA. Still, the mean level of repetition of MA was considerably larger than that obtained by ILS-MOEA/D, specifically, 25.4%. Bearing the above discussion in mind, it is clear that the slight difference obtained by ILS-MOEA/D, in terms of the cost, is much lower when compared to the difference between the two approaches in terms of the level of repetition. As a result, ILS-MOEA/D clearly outperforms MA in this regard, which confirms our hypothesis.

In order to graphically confirm the above conclusions, the best solution found by MA, i.e., that with the lowest cost, was compared to the first of the thirty Pareto Fronts obtained by ILS-MOEA/D for each instance considered. To do this, the level of repetition of the solutions obtained by MA had to be calculated independently by considering the same source code implemented in the case of the multi-objective constrained formulation of the MPP. Figure 2 undoubtedly shows how MA obtains the best results in terms of the menu plan cost, for every instance. Nevertheless, the level of repetition of said solutions is significantly higher when compared to the level of repetition of the solutions belonging to the Pareto Fronts provided by ILS-MOEA/D. Note also how ILS-MOEA/D yielded

different Pareto Front shapes, depending on the instance size. Bearing the above in mind, it is difficult to apply a priori methods to focus the search on a certain region of the Pareto Front. Finally, we note that one of the main advantages of applying a multi-objective optimiser, such as ILS-MOEA/D, is that a diverse set of solutions, regarding the objective function space, is provided. All of these solutions are trade-offs between cost and level of repetition, and the above allows the decision maker to select one of those solutions depending on their particular requirements, something that is not possible when solving a single-objective formulation of the MPP.

Table 3. Statistics for the cost and level of repetition achieved by ILS-MOEA/D and MA, considering menu plans for $n = 20$, $n = 40$ and $n = 60$ days.

n	Algorithm	Cost				Level of Repetition			
		Mean	Std	Min	Max	Mean	Std	Min	Max
20	ILS-MOEA/D	20.299	1.384×10^{-02}	20.264	20.304	269.800	3.695	263.166	278.222
	MA	20.264	7.226×10^{-15}	20.264	20.264	346.074	14.508	314.943	376.298
40	ILS-MOEA/D	40.153	0.102	40.038	40.335	601.309	8.616	579.400	615.871
	MA	40.067	0.042	40.038	40.281	833.067	33.055	757.831	888.489
60	ILS-MOEA/D	60.368	0.154	60.076	60.636	994.252	19.886	949.750	1029.680
	MA	59.738	0.045	59.721	59.940	1247.270	36.995	1143.820	1312.760

In the previous experiment, ILS-MOEA/D was unable to achieve the lowest cost provided by MA for a menu plan of $n = 60$ days in any execution. In order to determine if ILS-MOEA/D could yield the results attained by MA in large instances, such as $n = 60$, executions using a stopping criterion equal to ten hours were also performed. As in the previous experiment, 30 repetitions were also run by applying the same parameterisation of ILS-MOEA/D. The results obtained are shown in Table 4. We note that even though ILS-MOEA/D could not replicate the best menu plan cost obtained by MA after a five-hour execution (see Table 3), its results were improved in terms of both cost and level of repetition. As a result, devoting more effort to the development of ILS-MOEA/D could improve its performance when dealing with large instances.

Single-objective techniques that do not include any explicit diversity management mechanism may fall into premature convergence. This means that solutions only improve for a short period of time before stagnating at local optima. At the same time, multi-objective optimisers may also converge to sub-optimal regions of the decision variable space of some MOPs, yielding effects that are similar to premature convergence [42]. Although our proposal ILS-MOEA/D does not include any explicit technique for properly managing diversity in the decision variable space, it would be interesting to analyse if the above could be possible implicitly as a consequence of promoting diversity in the objective function space.

Table 4. Statistics for the cost and level of repetition achieved by ILS-MOEA/D, considering menu plans for $n = 60$ days, after ten hours of execution.

Cost				Repetition			
Mean	Std	Min	Max	Mean	Std	Min	Max
60.368	0.154	60.076	60.636	994.252	19.886	949.750	1029.680

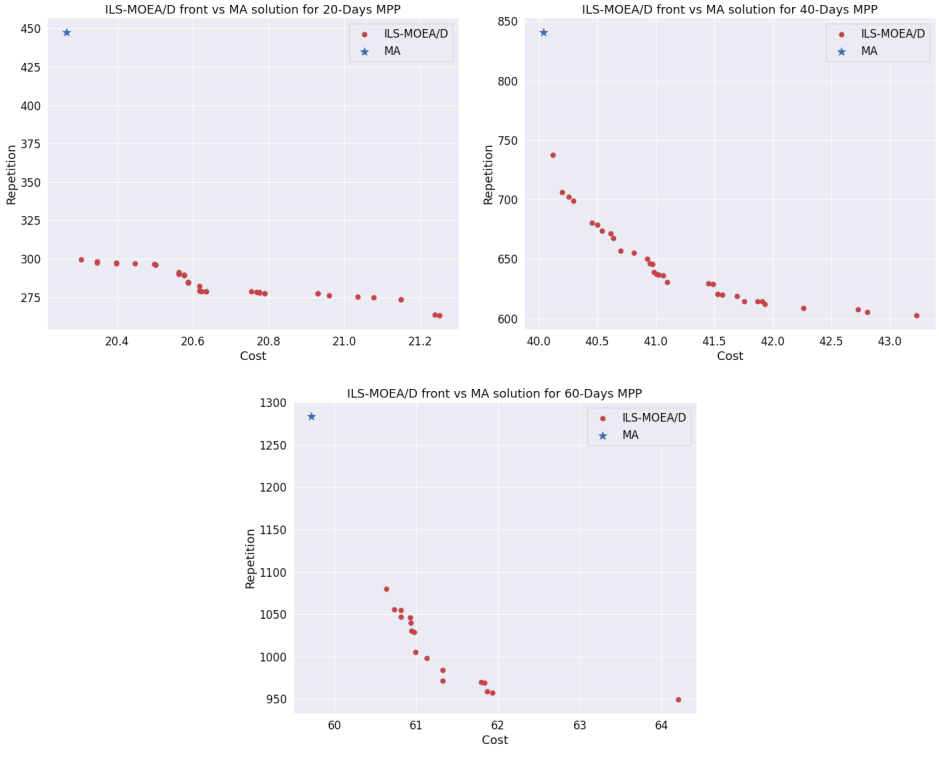


Figure 2. Comparison of the Pareto Fronts achieved by ILS-MOEA/D versus a two-dimensional representation of the best solution found by MA, i.e., the one with the lowest cost and lowest level of repetition, considering menu plans for $n = 20$ (**upper-left**), $n = 40$ (**upper-right**) and $n = 60$ (**bottom**) days. The reader should recall that the level of repetition of the solutions attained by MA had to be computed separately. To this end, the same source code implemented in the case of the multi-objective constrained formulation of the MPP was considered.

Figure 3 shows, in the case of MA and ILS-MOEA/D, how the mean diversity in the population, considering the decision variable space, evolves during the runs for each instance considered. Diversity was computed by applying a distance-like function specifically created for the MPP formulations considered herein. This function determines distances among the courses assigned to the different days of the plan. Further details about the distance-like function can be found in the reference work [3]. Now how the mean diversity gradually decreases throughout the executions for both MA and ILS-MOEA/D, starting from a more diverse population and finishing with one that is less diverse. Figure 4 shows the evolution of the mean HV over the course of the ILS-MOEA/D executions. Not only does the mean diversity gradually decrease throughout the executions for every instance in the case of ILS-MOEA/D, but the mean HV also increases, which means that the effectiveness of ILS-MOEA/D is suitable. Since ILS-MOEA/D implicitly promotes diversity in the objective function space, diversity is also properly managed in the decision variable space. At this point, we should note, however, that MA preserves, in a smarter and more explicit way, a larger diversity in the decision variable space during the whole execution in comparison to ILS-MOEA/D, which results in better cost solutions, something that we had already stated in previous experiments.

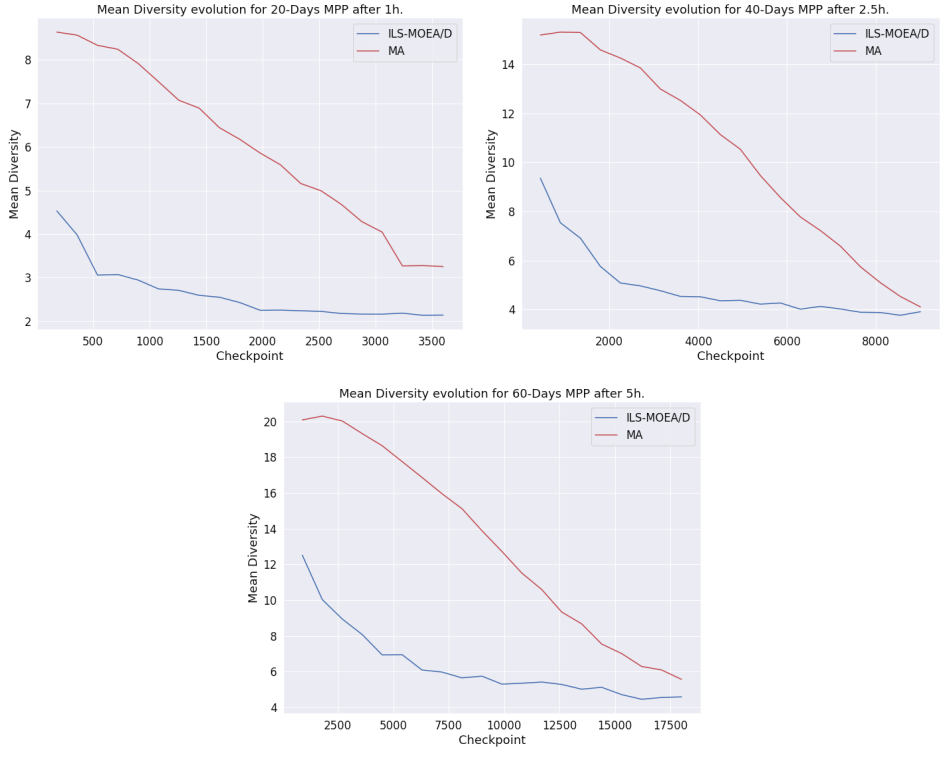


Figure 3. Evolution of the mean diversity of the population in the decision variable space, considering menu plans for $n = 20$ (**upper-left**), $n = 40$ (**upper-right**) and $n = 60$ (**bottom**) days.

Consequently, we note that not considering explicit diversity management schemes in the decision variable space in the case of a multi-objective algorithm, like ILS-MOEA/D, seems to impact the performance less than not considering them in the case of a single-objective optimiser. Even so, it would be interesting to check in the future whether explicitly managing diversity in the decision variable space helps ILS-MOEA/D to improve its performance in terms of the cost of the resulting menu plans.

Lastly, regarding the time complexity of the proposed algorithm, the average elapsed time and generations performed by ILS-MOEA/D are presented in Table 5. As it can be observed, ILS is the most computationally expensive step of ILS-MOEA/D, since it involves the majority of the total computational work. In particular, more than 99% of the total elapsed time of the algorithm is performed by ILS.

Table 5. Running time of ILS-MOEA/D.

Days	Avg. ILS Time (s)	Avg. Total Time (s)	Avg. Number of Generations
20	3591.6 (99.47%)	3610.7	187.2
40	8979.2 (99.31%)	9041.3	112.5
60	17962.1 (99.29%)	18089.7	110.1

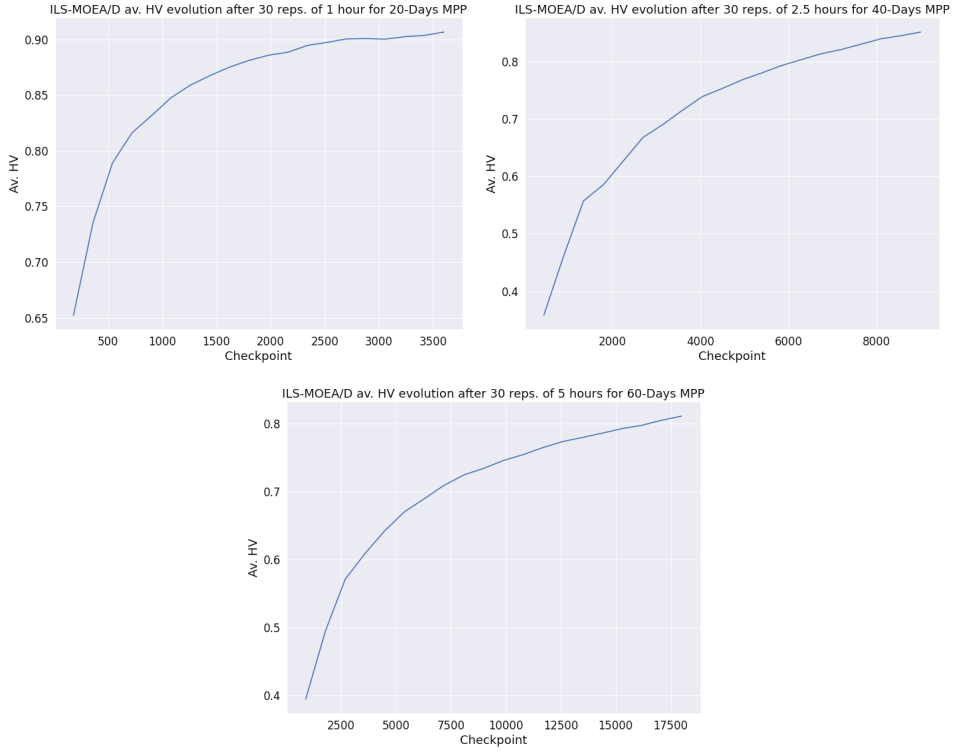


Figure 4. Evolution of the mean HV over the course of the executions, considering menu plans for $n = 20$ (**upper-left**), $n = 40$ (**upper-right**) and $n = 60$ (**bottom**) days.

6. Conclusions

This paper proposes a memetic multi-objective algorithm based on the well-known MOEA/D, which applies an ILS as the improvement phase, i.e., ILS-MOEA/D, to solve a novel multi-objective constrained formulation of the MPP. The experimental assessment conducted to contrast the diversity in the decision variable space of ILS-MOEA/D in comparison to a single-objective MA yielded interesting conclusions and possible future lines of work.

Firstly, depending on the problem size evaluated, ILS-MOEA/D obtained different Pareto Front shapes. This hinders the application of a priori methods to guide the search in a certain region of the space. Second, the results show that for reduced problem sizes, i.e., $n = 20$, ILS-MOEA/D and MA yield practically the same results in terms of the menu plan cost. However, as the problem size increases, such as $n = 40, 60$, the results provided by ILS-MOEA/D are slightly worse than those of MA in terms of the cost. Third, in the case of ILS-MOEA/D, diversity management in the decision variable space is not as crucial when compared to MA. Considering the results, both techniques gradually decrease the level of diversity; however, ILS-MOEA/D preserves lower levels of diversity in the same time even though MA includes an explicit diversity management technique. This is due to the fact that ILS-MOEA/D properly manages diversity in the decision variable space implicitly, since it also promotes diversity in the objective function space. Last but not least, the working hypothesis of this work is confirmed by the results provided in Section 5. Even though MA and ILS-MOEA/D yielded similar results in terms of the menu plan cost for the different instances assessed, the latter provided a much lower level of repetition of specific courses and food groups in comparison to the former. As a consequence, the application of ILS-MOEA/D to solve the multi-objective constrained formulation

of the MPP presented here, which considers the cost and the level of repetition as the objectives to be optimised, produces not only affordable, but also considerably more balanced, menu plans when compared to the plans obtained by MA when solving the single-objective formulation of the MPP, which only considers the cost.

As we previously mentioned, note that the single-objective MA incorporates a mechanism to explicitly promote diversity in the decision variable space, something that has not been included into ILS-MOEA/D. The above could be the main reason why ILS-MOEA/D obtained slightly worse meal plans in terms of their cost in comparison to the single-objective MA, particularly, for larger instances. Consequently, including techniques in ILS-MOEA/D to explicitly manage the diversity in the decision variable space could improve its performance in terms of the cost of the menu plans provided. Moreover, even though proposing a deep comparison of evolutionary algorithms is not aligned with our working hypothesis, further research may include a comparison with other multi-objective evolutionary algorithms and state-of-art metaheuristics. For example, it would be interesting to compare ILS-MOEA/D to differential evolution or particle swarm optimisation, even though major design changes would have to be considered to apply those algorithms to a combinatorial optimisation problem. As an alternative line of further research, it would be interesting to consider a comparison between ILS-MOEA/D and a single-objective MA intended to optimise the level of repetition of courses and food groups, rather than optimising the menu plan cost. Finally, we should note that our method could be applied to constrained multi-objective problems in other domains by performing a few modifications. First, the perturbation step of the ILS, as well as how neighbours are obtained, should be adapted by considering information of the particular problem at hand. Moreover, other problem-dependent variation operators should be incorporated into ILS-MOEA/D as well.

Author Contributions: Conceptualization, E.S., C.L. and C.S.; Formal analysis, A.M., E.S., C.L. and C.S.; Funding acquisition, A.M., E.S., C.L. and C.S.; Investigation, A.M., E.S., C.L. and C.S.; Methodology, E.S., C.L. and C.S.; Software, A.M., E.S. and C.S.; Supervision, E.S., C.L. and C.S.; Validation, E.S., C.L. and C.S.; Visualization, A.M.; Writing of original draft, A.M.; Writing of review & editing, A.M., E.S., C.L. and C.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially funded by the Spanish Ministry of Economy, Industry and Competitiveness as part of the programme "I+D+i Orientada a los Retos de la Sociedad" [contract number TIN2016-78410-R]. It was also partially funded by the Spanish Ministry of Science, Innovation and Universities, as well as by the University of La Laguna, as part of the programme "Nuevos Proyectos de Investigación: Iniciación a la Actividad Investigadora" [contract number 1203_2020]. The work of Alejandro Marrero was funded by the Canary Islands Government "Agencia Canaria de Investigación Innovación y Sociedad de la Información - ACIISI" [contract number TESIS2020010005]. The work was also funded by CONACyT through the "Ciencia Básica" project number 285599. Finally, the authors acknowledge the support from "Laboratorio de Supercómputo del Bajío" through project 300832 from CONACyT.

Acknowledgments: The problem formulation and constraints follow the recommendations of the school canteen programme of the Government of the Canary Islands: "Programa de Eco-comedores Escolares de Canarias", which is a joint priority action of the "Instituto Canario de Calidad Agroalimentaria", the "Dirección General de Salud Pública del Gobierno de Canarias" and the "Dirección General de Ordenación, Innovación y Promoción Educativa".

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Latasa, P.; Louzada, M.L.; Martinez Steele, E.; Monteiro, C. Added sugars and ultra-processed foods in Spanish households (1990–2010). *Eur. J. Clin. Nutr.* **2017**, *72*, 1404–1412. [CrossRef]
2. Ngo, H.C.; Cheah, Y.N.; Goh, O.S.; Choo, Y.H.; Basiron, H.; Kumar, Y.J.; Ngo, H.C.; Cheah, Y.N.; Goh, O.S.; Choo, Y.H.; et al. A review on automated menu planning approaches. *J. Comput. Sci.* **2016**, *12*, 582–596. [CrossRef]
3. Segura, C.; Miranda, G.; Segredo, E.; Chacon, J. A Novel Memetic Algorithm with Explicit Control of Diversity for the Menu Planning Problem. In Proceedings of the 2019 IEEE Congress on Evolutionary Computation, CEC 2019-Proceedings, Wellington, New Zealand, 10–13 June 2019; pp. 2191–2198.

4. Segredo, E.; Miranda, G.; Ramos, J.M.; León, C.; Rodríguez-León, C. SCHOOLTHY: Automatic Menu Planner for Healthy and Balanced School Meals. *IEEE Access* **2020**, *8*, 113200–113218. [CrossRef]
5. Gao, L.; Qian, W.; Li, X.; Wang, J. Application of memetic algorithm in assembly sequence planning. *Int. J. Adv. Manuf. Technol.* **2010**, *49*, 1175–1184. [CrossRef]
6. Rakshit, P.; Konar, A.; Bhowmik, P.; Goswami, I.; Das, S.; Jain, L.C.; Nagar, A.K. Realization of an adaptive memetic algorithm using differential evolution and Q-learning: A case study in multirobot path planning. *IEEE Trans. Syst. Man Cybern. Syst.* **2013**, *43*, 814–831. [CrossRef]
7. Shahidi, N.; Esmaeilzadeh, H.; Abdollahi, M.; Lucas, C. Memetic algorithm based path planning for a mobile robot. In *International Conference on Computational Intelligence*; Citeseer: State College, PA, USA, 2004.
8. Fahimnia, B.; Farahani, R.Z.; Sarkis, J. Integrated aggregate supply chain planning using memetic algorithm–A performance analysis case study. *Int. J. Prod. Res.* **2013**, *51*, 5354–5373. [CrossRef]
9. Zhang, Q.; Li, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **2007**, *11*, 712–731. [CrossRef]
10. Marrero, A.; Segredo, E.; Leon, C. On the automatic planning of healthy and balanced menus. In Proceedings of the GECCO 2019 Companion-Proceedings of the 2019 Genetic and Evolutionary Computation Conference Companion, Prague, Czech Republic, 13 July 2019.
11. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
12. Eckart Zitzler, M.L.; Thiele, L. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*; Technical Report; Swiss Federal Institute of Technology (ETH): Zurich, Switzerland, 2001.
13. Shir, O.M.; Preuss, M.; Naujoks, B.; Emmerich, M. Enhancing decision space diversity in evolutionary multiobjective algorithms. In *International Conference on Evolutionary Multi-Criterion Optimization*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 95–109.
14. Balintfy, J.L. Menu Planning by Computer. *Commun. ACM* **1964**, *7*, 255–259, [CrossRef]
15. Gazan, R.; Brouzes, C.M.; Vieux, F.; Maillot, M.; Lluch, A.; Darmon, N. Mathematical optimization to explore tomorrow's sustainable diets: a narrative review. *Adv. Nutr.* **2018**, *9*, 602–616. [CrossRef]
16. Moreira, R.P.C.; Wanner, E.; Martins, F.V.C.; Sarubbi, J.F.M. An Evolutionary Mono-Objective Approach for Solving the Menu Planning Problem. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
17. Baykasoğlu, A.; Taşkiran, D.; Akkoyun, H.G. Development of a menu planning desicion support system for mass catering with an application [Toplu beslenme i<iş in menü planlama karar destek sistemi gelitirilmesi ve uygulanmasi]. *J. Fac. Eng. Archit. Gazi Univ.* **2016**, *31*, 191–200.
18. Chávez, O.; Marchi, J.; Pozos, P. Nutritional Menu Planning: A Hybrid Approach and Preliminary Tests. *Res. Comput. Sci.* **2014**, *82*, 93–104. [CrossRef]
19. Funabiki, N.; Taniguchi, S.; Matsushima, Y.; Nakanishi, T. A Proposal of a Menu Planning Algorithm for Two-phase Cooking by Busy Persons. In Proceedings of the 2011 International Conference on Complex, Intelligent, and Software Intensive Systems, Seoul, Korea, 30 June–2 July 2011; pp. 668–673.
20. Isokawa, T.; Matsui, N. Performances in GA-based menu production for hospital meals. In Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC), Sendai, Japan, 25–28 May 2015; pp. 2498–2501.
21. Kashima, T.; Matsumoto, S.; Ishii, H. Evaluation of menu planning capability based on multi-dimensional 0/1 knapsack problem of nutritional management system. *IAENG Int. J. Appl. Math.* **2009**, *39*, 163–170.
22. Aberg, J. An Evaluation of a Meal Planning System: Ease of Use and Perceived Usefulness. In Proceedings of the 23rd British HCI Group Annual Conference on People and Computers: Celebrating People and Technology, Swinton, UK, 1–5 September 2009; pp. 278–287.
23. Gumustekin, S.; Senel, T.; Cengiz, M. A Comparative Study on Bayesian Optimization Algorithm for Nutrition Problem. *J. Food Nutr. Res.* **2014**, *2*, 952–958, [CrossRef]
24. Hernández-Ocaña, B.; Chávez-Bosquez, O.; Hernández-Torruco, J.; Canul-Reich, J.; Pozos-Parra, P. Bacterial Foraging Optimization Algorithm for Menu Planning. *IEEE Access* **2018**, *6*, 8619–8629. [CrossRef]
25. Hsiao, J.; Chang, H. SmartDiet: A personal diet consultant for healthy meal planning. In Proceedings of the 2010 IEEE 23rd International Symposium on Computer-Based Medical Systems (CBMS), Perth, WA, Australia, 12–15 October 2010; pp. 421–425, [CrossRef]

26. Kahraman, A.; Seven, H. Healthy Daily Meal Planner. In Proceedings of the 7th Annual Workshop on Genetic and Evolutionary Computation, Washington, DC, USA, 25 June 2005; ACM: New York, NY, USA, 2005; pp. 390–393, [CrossRef]
27. Moreira, R.; Wanner, E.; Martins, F.; Sarubbi, J. CardNutri: A Software of Weekly Menus Nutritional Elaboration for Scholar Feeding Applying Evolutionary Computation. In *International Conference on the Applications of Evolutionary Computation*; Springer: Cham, Switzerland, 2018; Volume 10784 LNCS, pp. 897–913, [CrossRef]
28. Seljak, B.K. Computer-based dietary menu planning. *J. Food Compos. Anal.* **2009**, *22*, 414–420. [CrossRef]
29. Seljak, B.K. Dietary Menu Planning Using an Evolutionary Method. In Proceedings of the 2006 International Conference on Intelligent Engineering Systems, London, UK, 26–28 June 2006; pp. 108–113.
30. Kaldrim, E.; Köse, Z. Application of a Multi-Objective Genetic Algorithm to the Modified Diet Problem. Ph.D. Thesis, Istanbul Technical University, Istanbul, Turkey, 2006.
31. Moreira, R.; Wanner, E.; Martins, F.; Sarubbi, J. The Menu Planning Problem: A Multiobjective Approach for Brazilian Schools Context. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Berlin, Germany, 15 July 2017; ACM: New York, NY, USA, 2017; pp. 113–114, [CrossRef]
32. Sufahani, S.; Ismail, Z. A new menu planning model for Malaysian secondary schools using optimization approach. *Appl. Math. Sci.* **2014**, *8*, 7511–7518. [CrossRef]
33. Eiben, A.E.; Smith, J.E. *Introduction to Evolutionary Computing*, 2nd ed.; Springer Publishing Company, Incorporated: Berlin, Germany, 2015.
34. Gaál, B.; Vassányi, I.; Kozmann, G. Application of artificial intelligence for weekly dietary menu planning. *Stud. Comput. Intell.* **2007**, *65*, 27–48.
35. Neri, F.; Cotta, C. Memetic algorithms and memetic computing optimization: A literature review. *Swarm Evol. Comput.* **2012**, *2*, 1–14. [CrossRef]
36. Chen, X.; Ong, Y.; Lim, M.; Tan, K.C. A Multi-Facet Survey on Memetic Computation. *IEEE Trans. Evol. Comput.* **2011**, *15*, 591–607. [CrossRef]
37. Ong, Y.S.; Lim, M.H.; Zhu, N.; Wong, K.W. Classification of adaptive memetic algorithms: a comparative study. *IEEE Trans. Syst. Man Cybern. Part B (Cybernetics)* **2006**, *36*, 141–152.
38. Ma, X.; Zhang, Q.; Tian, G.; Yang, J.; Zhu, Z. On Tchebycheff Decomposition Approaches for Multiobjective Evolutionary Optimization. *IEEE Trans. Evol. Comput.* **2018**, *22*, 226–244. [CrossRef]
39. León, C.; Miranda, G.; Segura, C. METCO: A Parallel Plugin-Based Framework for Multi-Objective Optimization. *Int. J. Artif. Intell. Tools* **2009**, *18*, 569–588. [CrossRef]
40. Segura, C.; Coello, C.; Segredo, E.; Aguirre, A.H. A Novel Diversity-Based Replacement Strategy for Evolutionary Algorithms. *IEEE Trans. Cybern.* **2016**, *46*, 3233–3246. [CrossRef]
41. Auger, A.; Bader, J.; Brockhoff, D.; Zitzler, E. Theory of the Hypervolume Indicator: Optimal μ-distributions and the Choice of the Reference Point. In Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms, Orlando, FL, USA, 9 January 2009; ACM: New York, NY, USA, 2009; pp. 87–102.
42. Castillo, J.C.; Segura, C.; Aguirre, A.H.; Miranda, G.; León, C. A Multi-Objective Decomposition-Based Evolutionary Algorithm with Enhanced Variable Space Diversity Control. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Berlin, Germany, 15 July 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 1565–1571.

mathematics

MDPI

Article

Comparison between Single and Multi-Objective Evolutionary Algorithms to Solve the Knapsack Problem and the Travelling Salesman Problem

Mohammed Mahrach, Gara Miranda *, Coromoto León and Eduardo Segredo

Departamento de Ingeniería Informática y de Sistemas, Universidad de La Laguna, Apto. 456, 38200 San Cristóbal de La Laguna, Tenerife, Spain; mmahrach@ull.edu.es (M.M.); cleon@ull.edu.es (C.L.); esegredo@ull.edu.es (E.S.)

* Correspondence: gmiranda@ull.edu.es

Received: 7 October 2020; Accepted: 9 November 2020; Published: 12 November 2020

Abstract: One of the main components of most modern Multi-Objective Evolutionary Algorithms (MOEAs) is to maintain a proper diversity within a population in order to avoid the premature convergence problem. Due to this implicit feature that most MOEAs share, their application for Single-Objective Optimization (SO) might be helpful, and provides a promising field of research. Some common approaches to this topic are based on adding extra—and generally artificial—objectives to the problem formulation. However, when applying MOEAs to implicit Multi-Objective Optimization Problems (MOPs), it is not common to analyze how effective said approaches are in relation to optimizing each objective separately. In this paper, we present a comparative study between MOEAs and Single-Objective Evolutionary Algorithms (SOEAs) when optimizing every objective in a MOP, considering here the bi-objective case. For the study, we focus on two well-known and widely studied optimization problems: the Knapsack Problem (KNP) and the Travelling Salesman Problem (TSP). The experimental study considers three MOEAs and two SOEAs. Each SOEA is applied independently for each optimization objective, such that the optimized values obtained for each objective can be compared to the multi-objective solutions achieved by the MOEAs. MOEAs, however, allow optimizing two objectives at once, since the resulting Pareto fronts can be used to analyze the endpoints, i.e., the point optimizing objective 1 and the point optimizing objective 2. The experimental results show that, although MOEAs have to deal with several objectives simultaneously, they can compete with SOEAs, especially when dealing with strongly correlated or large instances.

Keywords: multi-objective optimization; single-objective optimization; evolutionary algorithm; knapsack problem; travelling salesman problem

1. Introduction

Evolutionary Algorithms (EAs) [1] were initially developed for unconstrained Single-Objective Optimization Problems (SOPs). However, extensive research has been conducted to adapt them to other types of problems. In recent years, many Multi-Objective Evolutionary Algorithms (MOEAs) have been proposed in the literature [2,3] to adapt EAs to dealing with Multi-Objective Optimization Problems (MOPs). One of the main components of most modern MOEAs is the ability to maintain genetic diversity within a population of individuals [4]. Maintaining proper diversity is decisive for the behavior of EAs, since a loss of diversity could lead to premature convergence, which is a frequent drawback, especially for single-objective optimization. Most MOEAs implicitly manage diversity by considering the objective function space [5] and, in some cases, the decision variable space. Several mechanisms have been proposed in the literature to deal with the above, such as fitness

sharing [6], clustering [7], and entropy [8], among others [4]. Promoting diversity is a key feature of an efficient and reliable MOEA. In fact, it is an intrinsic component in many MOEAs. Because of this, some authors have claimed that the application of MOEAs might be useful when dealing with single-objective problems. Furthermore, several theoretical and empirical studies have shown that multi-objective optimizers can even provide better solutions than single-objective optimizers [4,9–11].

MOEAs have been applied to SOPs using various guidelines. Usually, the mechanisms proposed in the literature for solving SOPs by means of MOEAs consist of transforming the original SOP into a MOP so that MOEAs can be applied to the transformed problem. This transformation can be done by either replacing the original objective with a set of new objectives, or by adding new, additional objectives to the original one [4,12]. Among these approaches, the best known and most widespread in the literature are: transforming constraints into objectives [13], considering diversity as an explicit objective function [14] and multiobjectivization schemes, which transform a SOP into a MOP by modifying its fitness landscape [12]. In any case, these new objectives are included in order to promote the exploration of different regions, since multi-objective approaches try to simultaneously optimize several objectives. This might make it possible to escape from sub-optimal regions, thus providing a suitable balance between exploration and exploitation. The analysis presented in [15] lists the benefits of using additional objectives, named helper-objectives. The main ones are [12]: avoiding stagnation in local optima and maintaining diversity within a population.

In this paper, we present a comparative study of MOEAs and SOEAs when both types of schemes are separately applied to optimize each objective function of a bi-objective optimization problem. The study is not intended to provide a novel algorithm or to compare a new proposal with state-of-the-art algorithms. The main goal of this work is to investigate the effectiveness—or at least the opportunities—of applying multi-objective approaches to single-objective optimization. This study relies on comparisons of standard MOEAs and some general SOEAs when they seek to optimize—independently—every objective in a bi-objective problem. In this study, we consider the Knapsack Problem [16] and the Travelling Salesman Problem [17]. Both problems have been considered in numerous theoretical and experimental studies in the literature, so many effective solvers are known to perform successfully for a wide range of benchmarks.

Although there are many contributions that have been made in the field of mathematical optimization, in this work we are interested in the analysis of a particular set of approximated algorithms—named evolutionary algorithms—for both, single and multi-objective formulations of the problems. For this reason, our literature review deepens the field of evolutionary computation and not in other research areas that could also have great impact and interest nowadays. As an alternative, some experts have advocated for pushing further the integration of machine learning and combinatorial optimization [18]. Some operations research communities are introducing machine learning as a modeling tool for discrete optimization [19] or to extract intuition and knowledge in order to dynamically adapt the optimization process [20]. Despite the existence of such a huge amount of alternatives to face these optimization problems, it is important to note that we are interested in the comparative analysis of single and multi-objective evolutionary algorithms. Thus, the selected optimization problems can be understood as simple use cases for our experimental study.

For the experiments, we have selected an extensive and diverse set of problem instances that consider different features, sizes and complexities. However, all the instances have two optimization objectives, meaning they can be used to apply multi-objective approaches. For the optimization process, three MOEAs and two SOEAs have been analyzed. Each SOEA is applied twice for each problem instance (one for each objective), so that the optimized values for each of the two objectives can be compared to the multi-objective solutions offered by the MOEAs in question. The rest of this paper is organized as follows: Section 2 describes the formulation of the two problems selected for this study, as well as the set of instances solved during the experimental process. Then, Section 3 provides an overview of the approaches—MOEAs and SOEAs— applied during this study. A detailed description of the experimental analysis and some underlying results, as well as their discussion,

are presented in Section 4 Finally, the conclusions and some lines for future work are presented in Section 5.

2. Problems: Formulation and Instances

This section presents two well-known problems, the Knapsack Problem (KNP) [16] and the Travelling Salesman Problem (TSP) [17], which we have selected to conduct out experimental study. For each problem, a formulation involving two objectives is described, as well as the corresponding set of instances. Note that all instances presented below are bi-objective instances. This means that all instances have information that allow the calculation of two different objective functions to be carried out. Anyway, for the single-objective approaches, we can use only one of the objectives and discard the other, depending on the objective being analyzed at a particular moment.

2.1. The Bi-Objective Knapsack Problem (BOKNP)

We consider the one-dimensional 0/1 knapsack problem with two objectives, where fractional items are not allowed and each item is available only once. This multi-objective one-dimensional binary knapsack problem can be defined as follows. Given a set of items $J = \{1, ..., n\}$, each with an associated weight $w_j \in \mathbb{N}^*$ and a profit $c_j^k \in \mathbb{N}_0$ for each objective $k \in K = \{1, ..., p\}$, the problem seeks to select the subset of J whose total weight does not exceed a fixed capacity $W \in \mathbb{N}^*$, while simultaneously maximizing the accumulated profit according to each objective in K. Mathematically, the problem can be formulated as follows [21]:

$$\begin{array}{ll} \max f_1(x) & \sum_{j=1}^{n} c_j^1 x_j \\ \vdots & \vdots \\ \max f_p(x) & \sum_{j=1}^{n} c_j^p x_j \\ \text{subject to} & \sum_{j=1}^{n} w_j x_j \leq W,\ x_j \in \{0,1\} \end{array} \tag{1}$$

Moreover, and without loss of generality, we assume that $c_j^k \geq 0$ and $w_j \leq W : \forall j \in \{1, ..., n\}$, $\forall k \in \{1, ..., p\}$. Since, in this work, we are interested in a bi-objective formulation of this problem, we let $p = 2$, such that the set K contains two functions (f_1 and f_2) to be optimized, $K = \{1, 2\}$.

For this bi-objective knapsack problem, we propose using the subset benchmark "MOKP" data sets available in the MOCOlib project [22]. MOCOlib is a collection of data sets and links for a variety of multi-objective combinatorial optimization problems. In this collection, we found three different sets of instances that are suitable for the bi-objective 0/1 unidimensional knapsack problem defined herein.

The data files themselves contain a description of the instances. Table 1 is attached in order to summarize the main features of the different sets of instances as well as their original references. Some of the key points for each data set are briefly broken down here:

1. **Data set 1A**: consists of five data files (instances) for the bi-objective 0/1 unidimensional knapsack problem. The values for the profits and weights have been uniformly generated. The number of items in the instances range from 50 to 500. The tightness ratio (Equation (2)) is in the range $[0.11, 0.92]$.
$$r = \frac{W}{\sum_{i=1}^{n} w_i} \tag{2}$$
2. **Data set 1B**: consists of 40 data files (instances) corresponding to the 10 bi-objective 0/1 unidimensional knapsack problems. Every instance has a tightness ratio $r = 0.5$. For each problem, four variants (class A, B, C and D) are given:
    - **1B/A**: the weights and profits are uniformly distributed within the range $[1, 100]$.
    - **1B/B**: these instances are created starting from data set 1B/A by defining the objectives in reverse order.
    - **1B/C**: the profits are generated with plateaus of values of length $\leq 0.1 \times n$.

- **1B/D**: these instances are created starting from data set 1B/C by defining the objectives in reverse order.

3. **Data set 2**: consists of 50 data files (instances) that also correspond to the bi-objective 0/1 unidimensional knapsack problem. For each data set the value for W is computed as the nearest integer value of $(P/100)\sum_{j=1}^{n} w_j$ (where P is a percentage of $\sum_{j=1}^{n} w_j$). All these instances have a tightness ratio $r = 0.5$. Two types of correlated instances (WEAK and STRONG), as well as uncorrelated instances (UNCOR) were generated as follows [21]:

   - **UNCOR**: 20 uncorrelated instances of 50 items. The profit vectors c_j^1, c_j^2 and the weight vector w_j are uniformly generated at random in the range $[1, 300]$ for ten items, while for the remaining ones the range $[1, 1000]$ is considered.
   - **WEAK**: 15 weakly correlated instances ranging in size from 50 to 1000 items, where c_j^1 is correlated with c_j^2, i.e., $c_j^2 \in [111, 1000]$, and $c_j^1 \in [c_j^2 - 100, c_j^2 + 100]$. The weight values w_j are uniformly generated at random in the range $[1, 1000]$.
   - **STRONG**: 15 strongly correlated instances with the number of items ranging between 50 and 1000. The weights w_j are uniformly generated at random and are correlated with c_j^1, i.e., $w_j \in [1, 1000]$, and $c_j^1 = w_j + 100$. The value of c_j^2 is uniformly generated at random in the range $[1, 1000]$.

Table 1. Bi-objective KNP instances. The instance number (s), the number of items (n) and tightness ratio (r) refer to the parameters of the instances.

Set	Source	Name	Parameters
Set 1A	Gandibleux and Freville [23]	2KNP50-r	$n = 50$; $r \in \{0.11, 0.50, 0.92\}$
		2KNP100-50	$n = 100$; $r = 0.50$
		2KNP500-41	$n = 500$; $r = 0.41$
Set 1B/A	Visée et al. [24]	2KNPn-1A	$n \in \{100, 200, 300, 400, 500\}$; $r = 0.5$
Set 1B/B,C,D	Degoutin and Gandibleux [25]	2KNPn-1B	$n \in \{100, 200, 300, 400, 500\}$; $r = 0.5$
		2KNPn-1C	$n \in \{100, 200, 300, 400, 500\}$; $r = 0.5$
		2KNPn-1D	$n \in \{100, 200, 300, 400, 500\}$; $r = 0.5$
Set 2 (UNCORR)	Captivo et al. [21]	F5050Ws	$s \in \{01, 02, 03, ..., 10\}$; $n = 50$; $r = 0.5$
		K5050Ws	$s \in \{01, 02, 03, ..., 10\}$; $n = 50$; $r = 0.5$
Set 2 (WEAK)	Captivo et al. [21]	W4C50W01	$n = 50$; $r = 0.5$
		W4100W1	$n = 100$; $r = 0.5$
		4WnW1	$n \in \{150, 200, ..., 1000\}$; $r = 0.5$
Set 2 (STRONG)	Captivo et al. [21]	S1C50W01	$n = 50$; $r = 0.5$
		S1nW1	$n \in \{100, 150, 200\}$; $r = 0.5$
		1SnW1	$n \in \{250, 300, ..., 1000\}$; $r = 0.5$

2.2. The Bi-Objective Travelling Salesman Problem (BOTSP)

In this work we consider a generalization of the classical Travelling Salesman Problem (TSP), which is defined as follows. Given a complete graph—or fully connected network—$G = (V, E)$ with vertex set V (cities), edge set E (paths between any two cities $i, j \in \{1, ..., n\}$), and edge values c_{ij}^k with $k \in K = \{1, ..., p\}$ (objective cost—it could be distance, time, energy, etc.—between city i and city j), the problem is to find the Hamiltonian path [26] (tour), which is a single and cyclic circuit, along the edges of G, such that each vertex (city) is visited exactly once and the total tour for each objective k, defined as the sum of costs c_{ij}^k, is minimized. A more detailed description of this multi-objective formulation of TSP can be found in [27].

Given a graph $G = (V, E)$, where $V = \{1, 2, ..., n\}$ and $E = \{(i, \pi(i)), i \in V\}$, Π_n denotes the set of all possible permutations of n cities. For a permutation $\pi \in \Pi_n$, $\pi(i)$ represents the city that follows city i on the tour represented by permutation π. A permutation whose graph is a Hamiltonian path is

called a cyclic permutation. We denote by Π^c the set of all cyclic permutations of n cities. Therefore, a TSP tour can be represented by a permutation $\pi = (\pi(1), \ldots, \pi(n)) \in \Pi^c$. Thus, the formulation of the multi-objective TSP is given by:

$$\min_{\pi \in \Pi^c} \sum_{i=1}^{n-1} c^k_{\pi(i),\pi(i+1)} + c^k_{\pi(n),\pi(1)} \quad k = \{1, ..., p\} \tag{3}$$

Since in this work we are interested in multi-objective problems with two optimization objectives, we have considered the bi-objective TSP formulation. Thus, the general Equation (3) is considered, in which $k = \{1, 2\}$. Figure 1 is provided to better clarify the differences between a single and a bi-objective formulation of the TSP. Figure 1a illustrates a single-objective formulation of the TSP where there is only one set of costs (one for each edge), thus defining a single optimization function. As a result, the single-objective formulation of the TSP consists of a list of n cities and a set of costs—a single cost for each pair of cities—which are all stored in a cost matrix D with elements c_{ij}, with $i, j \in \{1, ..., n\}$, and diagonal elements $c_{ii} = 0$. However, Figure 1b shows the differences between a single and a bi-objective instance of the TSP. As shown in the example, a bi-objective formulation considers instances with two different costs for each edge: one cost for objective 1 and another for objective 2. Instead of having a single cost matrix, in a multi-objective formulation, we need to manage a cost matrix for each objective function considered.

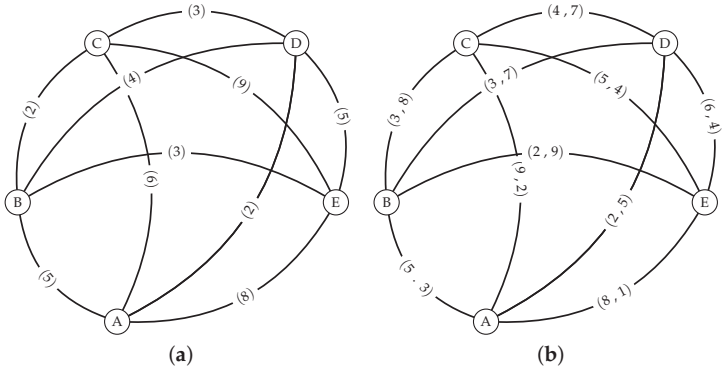


Figure 1. Illustration of single and bi-objective TSP graphs. (**a**) A graph with weights (distances) on its edges as a single-objective optimization problem. (**b**) A graph with weights (distances and times) on its edges as a bi-objective optimization problem.

For this bi-objective formulation of the problem, we need a suitable set of problem instances: different types, sizes and costs between cities. Two types of instances are selected for the experimental study that is presented in this work. First, in the Euclidean instances, the costs between edges correspond to the Euclidean distance between two points on a plane, randomly sampled from a uniform distribution. Meanwhile, in the clustered instances, the points are randomly clustered on a plane, and the costs between edges correspond to the Euclidean distance. Then, the bi-objective instances are obtained by combining a pair of single-objective instances. Table 2 shows the information for the 19 problem instances of symmetric bi-objective TSPs with 100, 300 and 500 cities (these instances are available at http://www-desir.lip6.fr/~lustt/). These instances have been used in several related works [28–30], so they have been successfully solved in the literature. In fact, their exact fronts were already published by K. Florios (optimal fronts are available at https://sites.google.com/site/kflorios/motsp). More details on the selected instances are given below:

- The **TSPLIB Euclidean Instances** [31] (files with prefix kro, from the authors Krolak/Felts/Nelson) consist of 13 instances with two objectives which are generated on the basis of the single-objective

TSP instances from TSPLIB [32] (Library of Traveling Salesman Problems). The TSPLIB is a library of sample instances for the TSP (and related problems) from various sources and with different features.

- The **DIMACS Clustered Instances** [33] (files with prefix clus) are three instances that have been created using the random instance generator available from the 8th DIMACS implementation challenge site (the generator is available at http://dimacs.rutgers.edu/archive/Challenges/TSP/index.html).
- The **DIMACS Euclidean Instances** [30] (files with prefix eucl) are a set of three instances which were also generated using the DIMACS code.

Table 2. Bi-objective TSP instances.

Name	Origin	Type	Source	Num. of Variables	Combinations
clusABn	DIMACS	Clustered	Lust et al. [33]	$n \in \{100, 300, 500\}$	clusAn and clusBn
euclABn	DIMACS	Euclidean	Paquete et al. [30]	$n \in \{100, 300, 500\}$	euclAn and euclBn
kroABn	TSPLIB	Euclidean	Paquete et al. [30]	$n \in \{100, 150, 200, 300, 400, 500, 750, 1000\}$	kroAn and kroBn
kroACn	TSPLIB	Euclidean	Paquete et al. [30]	$n \in \{100\}$	kroAn and kroCn
kroADn					kroAn and kroDn
kroBCn					kroBn and kroCn
kroBDn					kroBn and kroDn
kroCDn					kroCn and kroDn

3. Optimization Approaches

This section provides a description of all the algorithmic approaches selected, and thus considered in the experimental study. We also attempt to justify the selection and design decisions made.

3.1. Multi-Objective Evolutionary Algorithms

Evolutionary Multi-Objective Optimization (EMO) [34] is a collection of research, applications and algorithms in the field of Multi-Objective Optimization (MO) paradigms using Evolutionary Algorithms (EAs). In the related literature, several Multi-Objective Evolutionary Algorithms (MOEAs) have been proposed for solving MOPs, and these can be classified based on different features. A widely accepted classification for MOEAs is one that considers the following families:

- **Pareto-dominance-based algorithms** use the Pareto dominance relationship, where the partner of a non-dominated individual is chosen from among the individuals of the population that it dominates. Some widely known algorithms from this type are: Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [35], Strength Pareto Evolutionary Algorithm 2 (SPEA2) [36] and Pareto Envelope-based Selection Algorithm II (PESA-II) [37].
- **Decomposition-based algorithms** transform a MOP into a set of SOPs using scalarizing functions. The resulting single-objective problems are then solved simultaneously. Some examples of algorithms that fall under this approach are Multi-Objective Genetic Local Search algorithm (MOGLS) [38], Cellular Multi-Objective Genetic Algorithm (C-MOGA) [39] and Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) [40], as well as their many other variants.
- **Indicator-based algorithms** use an indicator function to assess the quality of a set of solutions, combining the degree of convergence and/or the diversity of the objective function space with a metric. These algorithms attempt to find the best subset of Pareto non-dominated solutions based on the performance indicator. Their many variants include: Indicator Based-Selection Evolutionary Algorithm (IBEA) [41], S-Metric Selection Evolutionary Multi-Objective Optimization Algorithm (SMS-EMOA) [42], Fast Hypervolume Multi-Objective

Evolutionary Algorithm (FV-MOEA) [43] and Many-Objective Metaheuristic Based on R2 Indicator (MOMBI-II) [44].

The No Free Lunch Theorem in optimization [45] states that any algorithm that searches for an optimal cost or fitness solution is not universally superior to any other algorithm. Therefore, for experimental studies, at least one algorithm of each type is usually selected as part of the state of the art. In this work, we apply a Pareto-dominance-based algorithm (NSGA-II), a decomposition-based approach (MOEA/D) and an indicator-based algorithm (SMS-EMOA), which guides the search by means of the hypervolume metric. A brief description of said multi-objective approaches is provided below:

- **NSGA-II** [35] is a generational genetic algorithm and is one of the most popular multi-objective optimization algorithms, having been widely and successfully applied in many real-world applications. It is one of the first multi-objective algorithms to introduce elitism, i.e., the elites of a population are given the opportunity to be carried to the next generation. It uses a fast non-dominated sorting procedure based on Pareto front ranking in an effort to promote convergence, meaning it emphasizes non-dominated solutions. In addition to the reasons given above, we have selected this algorithm because it uses an explicit diversity preservation mechanism (crowding distance).
- **MOEA/D** [40] is probably the most representative decomposition-based multi-objective algorithm. It processes a multi-objective problem by decomposing it into a set of single-objective subproblems and then performing a heuristic search in order to optimize—simultaneously and cooperatively—said subproblems. Generally, a MOEA needs to maintain diversity in its population to produce a set of representative solutions. MOEAs, such as NSGA-II, use crowding distances to maintain diversity. In MOEA/D, a MOP is decomposed into a number of scalar optimization subproblems. Different solutions in the current population are associated with different subproblems. The diversity among these subproblems will naturally lead to diversity in the population [40], which could reinforce the rationale for selecting this algorithm in the context of this study.
- **SMS-EMOA** [42] is an indicator-based algorithm that implements a special selection operator that combines the hypervolume metric with the concept of Pareto dominance. Since the hypervolume is a measure frequently applied for comparing the results of MOEAs, the underlying idea is to explicitly manage and maximize the dominated hypervolume within the optimization process. Hypervolume, which is also used for comparison purposes, measures convergence, as well as diversity. The SMS-EMOA keeps a population of non-dominated and dominated individuals at a constant size. Keeping only non-dominated individuals might lead to small or even single-membered populations, and thus to a crucial loss of diversity. To avoid losing diversity, defining a lower bound for the population size was suggested in [42]. These are the reasons that make SMS-EMOA a good candidate for this study, especially to test the effectiveness of the diversity of this algorithm to improve results in MOPs.

3.2. Single-Objective Evolutionary Algorithms

In the context of SOEAs, some of the most frequently used approaches are Evolution Strategies (ESs) and Genetic Algorithms (GAs). The main differences between these types of EAs lie in the calculation of the fitness and the application of operators (mutation, recombination and selection). In contrast to GAs, where the main role of the mutation operator is simply to avoid the problem of premature convergence, mutation is the primary operator of ESs. Furthermore, in contrast to GAs, selection in the case of ESs is absolutely deterministic. For the experimental study conducted in this work, we considered the following approaches:

- **Generational Genetic Algorithm (gGA)** [46]: two parents are selected from the population in order to be crossed, yielding two offspring, which are later mutated and evaluated.

These newly generated individuals are placed in an auxiliary population that will replace the current population when it is completely full.

- **Steady-State Genetic Algorithm (ssGA)** [47]: two parents are selected and crossed, yielding two offspring that are later crossed. Then one of the resulting offspring is mutated. The mutated individual is evaluated and then inserted into the population, usually replacing the worst individual in the population (if the new one is better). Hence, the parents and offspring can co-exist in the population for the next iteration.
- **Elitist Evolution Strategy (μ + λ) (eES)** [48]: the elitist feature allows for the best solution to be always kept. The algorithm starts with a population of size μ. Each generation λ of mutated individuals is created from the current population. After the generation of the mutated individuals, there are a total of $(\mu + \lambda)$ individuals, including the parents and the new individuals generated from them. From these $(\mu + \lambda)$ individuals, the best μ ones are kept—as parents—for the next generation.
- **Non-Elitist Evolution Strategy (μ, λ) (neES)** [48]. in this case the best μ mutated individuals from among the new generated λ are selected as parents for the next generation, i.e., none of the μ parents survive the next generation, meaning $\lambda \geq \mu$ must hold.

3.3. Comparison of Single and Multi-Objective Approaches

For the comparison we will use the same set of bi-objective instances for the single-objective and multi-objective algorithms here considered. Our aim will be to analyze the objectives independently, i.e., first comparing the values of MOEAs and SOEAs for objective 1 in the whole set of instances considered, and then, in a similar way, by comparing objective 2. The multi-objective approaches directly address the bi-objective instances selected for the KNP and the TSP. The above means that they obtain, at every execution, an extreme—best—value for objective 1, and another extreme—best—value for objective 2. We note that both values—for the two optimization objectives—are obtained at the same time, i.e., in one single execution of the algorithm. However, the single-objective approaches cannot deal with several objectives simultaneously, and therefore, they need to be executed twice: once to optimize objective 1 and another to optimize objective 2.

During the experimental evaluation we will focus on solution quality when comparing the different approaches, i.e., we will not perform any analysis on the execution time for each approach. Due to their stochastic nature, the time complexity analysis of EAs is not an easy task [49]. Many experimental results have been reported on all types of EAs but only a few results have been proved on a theoretical context [50]. Besides, when the complexity analysis is about MOEAs, the development of a theoretical study is even more complicated [51]. Since MOEAs implicitly deal with objectives that are in conflict one with each other, they need to manage a set of trade-off solutions instead of one single (optimal) solution. When tackling MOPs is necessary to distinguish the quality of solutions consisting of multiple objective values. In many MOEAs, it is common to use the concepts of Pareto dominance in order to sort a set of solutions: non-dominated sorting. This sorting procedure aims to divide a solution set into a number of disjoint subsets or ranks, by means of comparing their values of the same objective. After the sorting process, solutions in the same rank are viewed equally important, and solutions in a smaller rank are better than those in a larger rank. Since a wide range of the existing MOEAs have adopted this sorting strategy, they all involve a high computational cost [52]. Some studies have shown that in an approach such as NSGA-II applied to a bi-objective DTLZ1 benchmark problem, the non-dominated sorting consumes more than 70% of the run-time for a population size of 1000 individuals and a maximum number of generations equal to 500 [52]. In our study, the execution times of the multi-objective approaches range from 5 to 10 times greater in comparison to the time required by the two executions–one for each objective–of the corresponding single-objective alternatives. These values depend on the problem (KNP or TSP) and on the instance type or size. When dealing with many-objective optimization problems (three or more objectives) this aspect of efficiency becomes even more critical. Bearing the above in mind, some authors have actively

worked on reducing the number of objective functions by eliminating those that are not essential to describe the Pareto-optimal front [53].

4. Experimental Results

As noted in previous sections, for the experimental study we considered the bi-objective KNP and TSP formulations. Moreover, we have described the set of instances that could be used in the context of these formulations. Regarding the type of approaches to apply, as mentioned previously, we are interested in evaluating the possibilities offered by a multi-objective optimization mechanism where we analyze, from the resulting Pareto front, the end points in each objective independently. The optimization approaches compared herein consist of three MOEAs (NSGA-II, MOEA/D, and SMS-EMOA) and two single-objective algorithms (gGA and eES). Note that, initially, we checked the behavior of four single-objective approaches, but two of them (ssGA and neES) were discarded for the exhaustive study presented here. This is because the results output by these algorithms were not at all competitive when compared to the single-objective algorithms finally selected for our comparisons (gGA and eES).

4.1. Parameter Setting

Since our focus is to conduct an experimental comparison between different MOEAs and different SOEAs, it was necessary to carry out an exhaustive process to adjust and analyze the ideal parameters for each algorithm. This section provides all the details on the algorithm configurations and the experimental set-up. It is important to note that all the algorithms were implemented in Java using the jMetal [54] framework (the source code used in the current work, as well as the results and graphics extracted from them, can be found through https://github.com/Tomas-Morph/knp-tsp-journal-mathematic). We also used the irace package [55] to set the automatic parameters in all of the algorithms implemented. For each problem—KNP and TSP—we defined a common solution encoding for all the algorithms implemented. We also decided to apply some standard and basic operators for all the algorithms implemented (and in the same way for all of them). To set the automatic parameters, a personalized adjustment was made for each approach. The set of configuration parameters that were automatically tuned—for each algorithm—are as follows:

- Common operator parameters: mutation and crossover probabilities.
- Algorithm parameters: population sizes and other algorithm-specific parameters.
- Other parameters: selection, crossover, and mutation operators. These parameters were set for each optimization problem, using the same operators for all the single and multi-objective approaches.

As previously indicated, the parameters listed were automatically tuned using the irace package. We first ran irace with the set of input parameters described in Table 3. For this initial tuning process, we selected a subset of representative instances (different type and sizes) for each problem. The best configuration obtained by this automatic process for each pair problem-algorithm after training for a few hours is shown in Table 4. Considering these parameter settings and in order to achieve statistically significant results, a total of 100 independent runs were executed for each pair (algorithm, problem instance) . In order to statistically support the conclusions, the following statistical testing procedure, which was used in a previous work by the authors [56], was applied to compare the results obtained by the different algorithmic schemes. First, a Shapiro–Wilk test was performed to check whether the values of the results followed a normal (Gaussian) distribution. If so, the Levene test checked for the homogeneity of the variances. If the samples had equal variance, an ANOVA test was done; if not, a Welch test was performed. For non-Gaussian distributions, the non-parametric Kruskal–Wallis test was used. For all the tests, a significance level of $\alpha = 0.05$ was considered.

Finally, it is important to note that this study is not intended to offer a comparison of the best-performing algorithms existing in the related literature; the main goal is to analyze the suitability of MOEAs for optimizing single-objective problems.

Table 3. Input parameter set for irace auto-configuration.

Input parameter	Possible values	NSGA	MOEA/D	SMS-EMOA	gGA	eES
Crossover probability	[0.0, 1.0]	✓	✓	✓	✓	
Mutation probability	[0.0, 1.0]	✓	✓	✓	✓	✓
Population size	{10, 20, 50, 100, 200, 300}	✓	✓	✓	✓	(μ)
Offspring population size	{1, 2, 5, 10, 20, 50, 100, 200, 300}	✓			✓	(λ)
Selection tournament size	[2, 10]	✓		✓	✓	
Hypervolume offset	{10, 20, 50, 100, 200, 500}			✓		
Neighborhood size	{10, 20, 50, 100}		✓			
Neighbor select probability	[0.0, 1.0]		✓			

Table 4. Parameter settings for each problem-algorithm pair.

Parameter	KNP					TSP				
Encoding	Binary strings					Permutation of integers				
Initial solutions	random					random				
Mutation operator	Bit-flip					Permutation swap				
Crossover operator	Single point					PMX				
Selection	Tournament					Tournament				
	NSGA	**MOEA/D**	**SMS-EMOA**	**gGA**	**eES**	**NSGA**	**MOEA/D**	**SMS-EMOA**	**gGA**	**eES**
Crossover probability	0.9784	0.9578	0.9512	0.8795	_	0.9843	0.9421	0.9754	0.7895	_
Mutation probability	0.0485	0.0578	0.0358	0.0081	0.2239	0.0163	0.0105	0.0093	0.1116	0.3806
Population size	20	200	20	20	1	20	300	20	20	1
Offspring population size	20	_	_	20	4	20	_	_	100	2
Tournament size	5	_	2	2	_	6	_	2	4	_
Hypervolume offset	_	_	200	_	_	_	_	200	_	_
Neighborhood size	_	20	_	_	_	_	50	_	_	_
Neighbor probability	_	0.8895	_	_	_	_	0.9354	_	_	_

4.2. Performance

The first set of experiments focused on studying how the algorithms evolved over the course of the executions. Figures 2 and 3 show the evolution—over the number of evaluations—of the mean fitness values for both objectives, for the KNP and TSP, respectively. For this set of experiments, the stopping criterion was set to a large number of function evaluations in order to analyze the convergence of the different approaches studied. This will allow us to set the stopping criteria for all the approaches and instances in subsequent experiments. Finally, we note that this preliminary experiment was not applied to the complete set of instances. Instead, a representative set of instances of different types and sizes was chosen for this preliminary overview.

For the KNP (see Figure 2), we solved a total of nine instances: three instances of size 50 and type UNCOR, three instances of different sizes (300, 600, 900) and type WEAK; and finally three of type STRONG with sizes of 300, 600, and 900. Each set of three instances of the same type (UNCOR, WEAK, or STRONG) is shown in the same row. For each instance, two graphs are shown; at the top, the one for objective 1, and the one for objective 2 below it. Note that, in most cases, the algorithms converge quickly for the initial evaluations. The convergence is only slower for the last instances, i.e., those that are strongly correlated. In general, we can see that all the algorithms yield a sharp increase in solution quality in the early generations of the search. We note that, from approximately $50 \cdot 10^3$ evaluations, the difference in performance remains constant during almost the entire run; as a result, we used this point as the stopping criterion for the next experiment.

Based on the behavior among the different instances, we can state the following. For the uncorrelated instances (first two rows; six graphs in total), we note that the SOEAs dominate for both objectives at all times, although the MOEAs are very close, with a relatively constant difference. However, for the weakly correlated instances, there is no apparent difference among the approaches, although the gGA appears to be slightly superior. Finally, for the strongly correlated instances, we see a clear dominance of the MOEAs for objective 1, with a notable difference between eES and the remaining approaches.

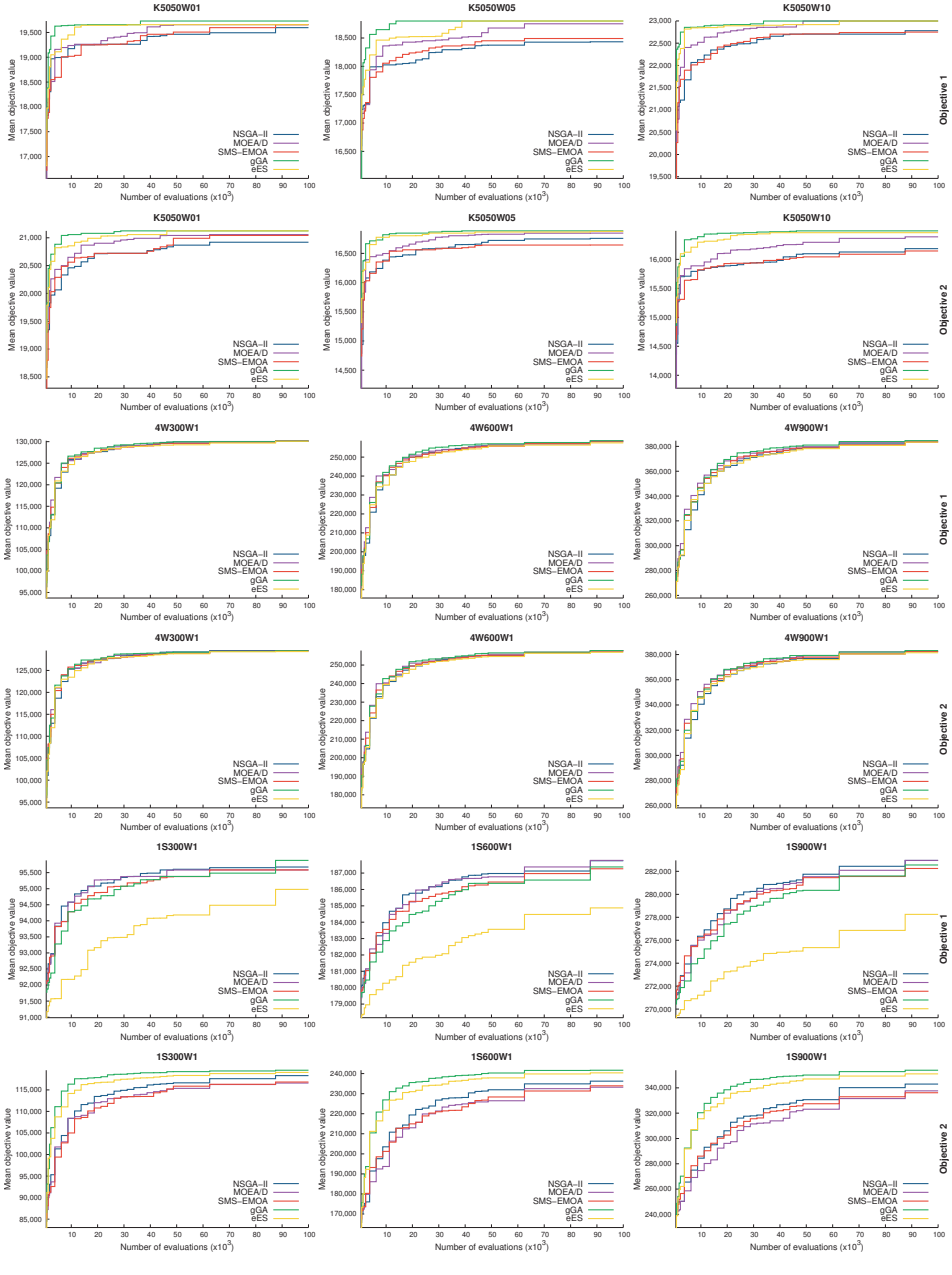


Figure 2. KNP evolution of the mean fitness for objectives 1 and 2.

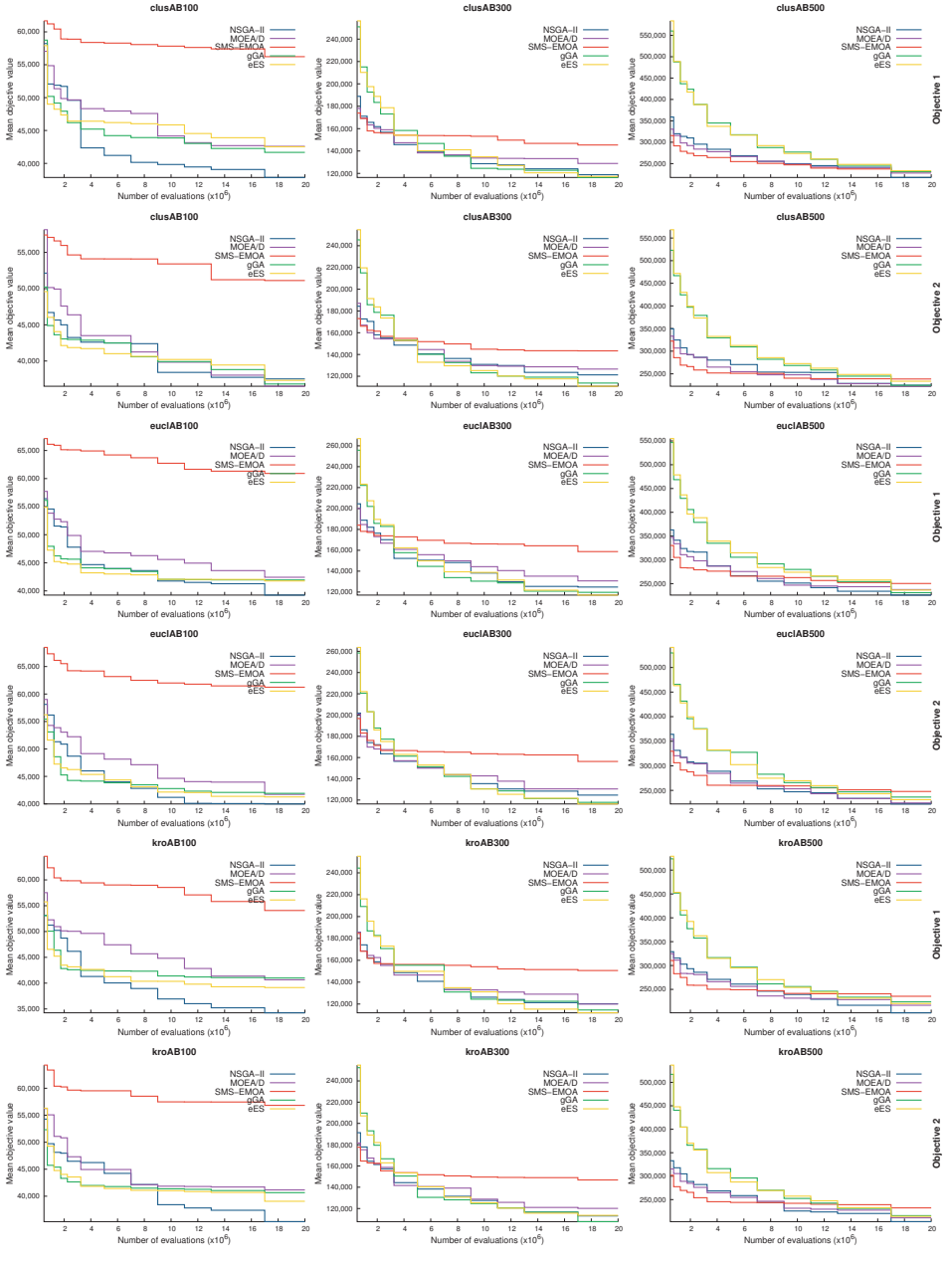


Figure 3. TSP evolution of the mean fitness for objectives 1 and 2.

For the TSP (see Figure 3), we notice that the size of the instances is directly related to the behavior of the algorithms: regardless of the type of instance and the objectives, we can differentiate three types of behaviors. For the small instances (with 100 cities), the SOEAs predominate at the beginning of the runs, but we see how NSGA-II always achieves better objective values from the middle of

the runs until the end. There is also a noticeable difference in SMS-EMOA, which stagnates from the beginning and fails to converge in all the small instances. However, in the case of medium-size instances (with 300 cities), the SMS-EMOA converges rapidly, together with NSGA-II, exhibiting better performance and surpassing the other algorithms, but only up to $2 \cdot 10^6$ evaluations approximately, where again SMS-EMOA stagnates and is overtaken by SOEAs, which eventually outperforms the other algorithms. For large instances (with 500 cities), once more, SMS-EMOA converges very quickly, in this case accompanied by the other two MOEAs, NSGA-II and MOEA/D, until almost the end of the runs, by which point the SOEAs manage to catch up to the other algorithms. Finally, as concerns the convergence and stagnation of the approaches, we have set the stopping criterion of subsequent experiments to $10 \cdot 10^6$ evaluations, in the case of the TSP.

Since, as we noted, the performance of the MOEAs in the TSP improves with the instance size—especially for SMS-EMOA—we decided to run the two largest instances in the TSP data set. These instances have 750 and 1,000 cities. Figure 4 shows that, for both objectives, MOEAs were able to provide better mean objective values than SOEAs during the entire run. In particular, SMS-EMOA yields the best results, despite being the algorithm that obtained the worst results in the small instances.

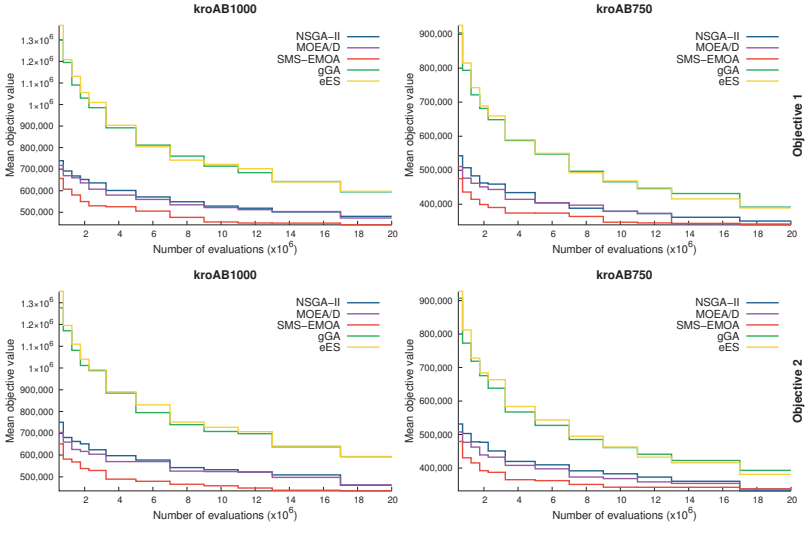


Figure 4. TSP (large instances) evolution of the mean fitness for objectives 1 and 2.

4.3. Optimization Behavior

Based on the previous results, for this experiment, we selected $50 \cdot 10^3$ and $10 \cdot 10^6$ function evaluations as the stopping criteria for the KNP and the TSP, respectively. As shown before, since the solutions stop improving by that point for any of the approaches, we can reduce the computational effort without losing generality in the analysis to be performed. Moreover, in this second experiment, we ran the complete set of instances and did the comparison at the end of the executions, once the corresponding stopping criterion was reached. Note that all the algorithms were executed 100 times. In order to compare the results obtained by the multi-objective approaches with those achieved by the single-objective optimizers, we calculated the extreme solutions in the Pareto optimal set, which correspond to the best solution attained for each objective function.

The results shown in Tables 5 and 6 correspond to the KNP problem, considering objective 1 and objective 2, respectively. Similarly, the results in Tables 7 and 8 correspond to the TSP. To facilitate the analysis, the mean and median solution values for each problem-instance-algorithm have been normalized as relative measures. Such relative solution values are expressed as a percentage with

respect to the best corresponding solution. For each problem instance we fixed the best solution as the best value found for a particular objective across the complete set of related executions. For each instance and objective, we have performed a total of 500 executions (5 algorithms $\times$ 100 executions each). From this total of 500 values, we fixed the best one as our reference to calculate the percentage of solution quality obtained by each proposal (as shown in the tables). Furthermore, for each instance, the cells containing the best median results have a gray background. Finally, the last column shows, for each instance considered, whether statistically significant differences arose when comparing the best-performing multi-objective approach against the best single-objective method by using the statistical comparison procedure described at the beginning of this section. The best-performing schemes are those that exhibit the best mean and median of the objective function for each test case. If any statistically significant differences exist, i.e., the p-value obtained from the statistical comparison procedure is lower than the significance level, an 'S' if shown if the corresponding single-objective algorithm provides a better mean and median of the corresponding objective function. If the best mean and median are provided by the corresponding multi-objective approach, an 'M' is shown. Finally, for those test cases where the two algorithms exhibit no statistically significant differences, a '-' is shown.

Table 5. Results for KNP instances (objective 1).

	Problem	Multi-Objective						Single-Objective				Test
		NSGAII		MOEA/D		SMSEMOA		GA		ES		
		Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	
Set 1/A	2KP100-50	76.1%	76.1%	81.1%	81.6%	77.3%	77.8%	96.1%	96.3%	93.5%	93.6%	**S**
	2KP50-11	85.3%	87.5%	89.0%	91.3%	88.9%	88.4%	99.1%	100.0%	96.3%	100.0%	**S**
	2KP50-50	83.4%	84.9%	92.0%	93.3%	84.0%	85.1%	97.8%	98.3%	95.4%	97.0%	**S**
	2KP50-92	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	99.7%	100.0%	98.0%	100.0%	**M**
	2KP500-41	63.1%	63.4%	63.3%	63.6%	62.4%	62.4%	94.3%	94.4%	88.9%	88.9%	**S**
Set 1/B	2KP100-1A	70.2%	70.9%	78.4%	78.0%	73.4%	73.5%	96.1%	96.5%	90.7%	90.6%	**S**
	2KP100-1B	70.1%	70.2%	80.1%	79.8%	72.5%	72.7%	96.7%	96.6%	90.4%	90.8%	**S**
	2KP100-1C	82.1%	82.6%	85.4%	85.4%	82.6%	83.3%	96.5%	96.6%	93.1%	93.3%	**S**
	2KP100-1D	83.1%	84.4%	88.0%	87.8%	83.8%	84.8%	96.5%	97.0%	93.4%	93.8%	**S**
	2KP200-1A	69.6%	70.2%	76.4%	76.6%	71.3%	71.7%	95.7%	95.9%	89.2%	89.6%	**S**
	2KP200-1B	70.7%	70.7%	76.7%	76.6%	71.6%	71.8%	96.1%	96.4%	90.2%	90.4%	**S**
	2KP200-1C	65.7%	65.9%	77.2%	77.3%	66.6%	67.3%	96.5%	96.4%	91.0%	91.2%	**S**
	2KP200-1D	65.2%	65.7%	73.4%	73.7%	67.8%	68.6%	96.4%	96.1%	90.6%	90.8%	**S**
	2KP300-1A	68.4%	68.4%	73.1%	73.6%	68.7%	69.1%	95.4%	95.4%	89.7%	90.1%	**S**
	2KP300-1B	69.8%	70.3%	73.7%	73.4%	68.8%	68.9%	95.9%	96.0%	90.5%	90.8%	**S**
	2KP300-1C	71.1%	71.7%	81.7%	82.2%	71.5%	71.4%	95.2%	95.5%	90.9%	91.2%	**S**
	2KP300-1D	80.5%	80.3%	82.2%	82.4%	77.3%	77.4%	95.1%	95.3%	89.9%	90.1%	**S**
	2KP400-1A	68.0%	68.2%	69.4%	69.1%	67.3%	67.7%	95.8%	96.1%	87.8%	87.9%	**S**
	2KP400-1B	66.2%	65.9%	68.2%	68.0%	66.1%	66.5%	95.5%	95.3%	87.5%	87.4%	**S**
	2KP400-1C	65.6%	65.2%	69.6%	69.2%	65.4%	65.1%	97.4%	97.6%	85.3%	85.8%	**S**
	2KP400-1D	55.6%	55.7%	59.2%	59.8%	56.3%	56.4%	96.9%	96.9%	87.1%	87.0%	**S**
	2KP500-1A	63.9%	64.4%	67.2%	66.8%	63.3%	63.5%	93.4%	93.7%	85.4%	85.4%	**S**
	2KP500-1B	64.0%	63.8%	66.8%	66.6%	62.7%	62.4%	94.1%	94.2%	86.6%	87.0%	**S**
	2KP500-1C	75.8%	75.9%	81.4%	81.7%	74.4%	74.7%	95.2%	95.1%	88.3%	88.5%	**S**
	2KP500-1D	72.3%	72.7%	71.3%	71.4%	69.3%	69.7%	95.0%	94.9%	88.8%	88.3%	**S**
Set 2/UNCOR	F5050W01	82.9%	84.0%	90.7%	91.9%	80.7%	83.3%	96.9%	97.7%	94.0%	93.7%	**S**
	F5050W02	90.4%	90.9%	94.8%	94.8%	86.3%	87.0%	99.0%	99.3%	97.2%	98.2%	**S**
	F5050W03	89.4%	90.0%	95.6%	96.4%	84.2%	82.2%	99.5%	100.0%	97.9%	100.0%	**S**
	F5050W04	94.1%	95.3%	96.1%	95.3%	89.4%	92.4%	98.8%	99.5%	97.2%	98.8%	**S**
	F5050W05	86.2%	84.6%	91.1%	93.4%	85.1%	84.6%	98.6%	100.0%	95.2%	94.2%	**S**
	F5050W06	87.9%	88.1%	93.1%	92.6%	84.7%	85.7%	97.1%	97.0%	93.8%	93.8%	**S**
	F5050W07	88.6%	89.9%	93.8%	94.9%	86.7%	87.8%	99.1%	99.7%	97.9%	97.9%	**S**
	F5050W08	90.0%	91.8%	94.0%	94.2%	83.7%	83.9%	98.6%	99.3%	94.9%	97.4%	**S**
	F5050W09	97.2%	98.9%	98.8%	99.2%	95.5%	96.5%	99.0%	99.5%	98.8%	99.5%	**S**
	F5050W10	93.3%	94.6%	97.2%	98.6%	90.2%	89.9%	98.8%	100.0%	97.4%	100.0%	**S**
	K5050W01	87.0%	88.2%	92.0%	93.9%	80.3%	82.1%	95.5%	94.4%	92.6%	93.9%	**S**
	K5050W02	88.0%	88.1%	91.6%	93.1%	83.5%	84.1%	98.6%	99.2%	95.0%	95.5%	**S**
	K5050W03	96.3%	97.9%	97.7%	98.5%	94.4%	94.4%	99.1%	99.0%	96.7%	97.9%	**S**
	K5050W04	92.4%	93.0%	95.9%	97.4%	86.5%	86.5%	99.5%	99.7%	97.0%	99.1%	**S**
	K5050W05	79.8%	79.9%	89.2%	88.1%	76.7%	76.8%	98.7%	100.0%	93.5%	91.5%	**S**
	K5050W06	87.4%	87.0%	93.5%	93.7%	83.5%	84.3%	97.5%	97.0%	96.2%	96.0%	**S**
	K5050W07	84.6%	85.3%	92.5%	94.1%	79.0%	80.3%	98.8%	98.9%	97.6%	98.4%	**S**
	K5050W08	93.0%	91.3%	93.9%	94.3%	89.7%	90.9%	96.6%	97.2%	95.3%	97.2%	**S**
	K5050W09	91.3%	91.6%	94.4%	94.2%	85.8%	87.8%	98.6%	99.4%	96.1%	96.1%	**S**
	K5050W10	87.8%	87.6%	95.2%	95.6%	81.5%	82.5%	97.9%	97.8%	97.4%	97.4%	**S**

Table 5. *Cont.*

	Problem	Multi-Objective						Single-Objective				Test
		NSGAII		MOEA/D		SMSEMOA		GA		ES		
		Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	
Set 2/WEAK	4W150W1	98.2%	98.3%	98.2%	98.3%	98.1%	98.1%	98.5%	98.5%	97.8%	97.8%	**S**
	4W1W1	94.1%	94.1%	96.3%	96.5%	95.2%	95.3%	97.7%	97.7%	93.8%	93.9%	**S**
	4W200W1	97.8%	97.9%	97.7%	97.7%	97.7%	97.7%	98.1%	98.2%	96.9%	97.1%	-
	4W250W1	90.9%	90.6%	91.1%	91.5%	89.6%	89.7%	92.9%	92.9%	88.1%	88.3%	**S**
	4W300W1	92.7%	92.7%	93.9%	94.1%	91.7%	92.1%	95.3%	94.9%	89.9%	90.1%	**S**
	4W350W1	91.9%	92.0%	93.2%	93.0%	91.4%	91.5%	95.8%	96.1%	90.0%	90.2%	**S**
	4W400W1	92.0%	92.1%	93.6%	94.2%	91.9%	91.8%	96.5%	96.8%	89.2%	88.8%	**S**
	4W450W1	88.0%	88.1%	90.1%	90.0%	88.2%	88.2%	93.2%	93.5%	85.9%	85.8%	**S**
	4W500W1	88.9%	88.9%	91.3%	91.8%	89.4%	89.5%	95.0%	95.3%	87.2%	87.5%	**S**
	4W600W1	86.5%	86.9%	89.9%	89.8%	87.5%	87.5%	93.3%	93.2%	84.5%	84.1%	**S**
	4W700W1	84.0%	84.0%	87.2%	87.3%	85.7%	85.8%	91.3%	91.5%	81.5%	81.7%	**S**
	4W800W1	84.9%	84.7%	89.7%	89.7%	87.4%	87.9%	93.6%	93.8%	83.6%	83.6%	**S**
	4W900W1	83.3%	83.5%	89.3%	89.4%	86.9%	86.9%	93.0%	93.4%	83.1%	83.2%	**S**
	W4100W1	94.2%	94.5%	94.4%	94.8%	93.5%	93.7%	94.7%	95.0%	92.2%	92.6%	-
	W4C50W01	98.9%	98.8%	99.1%	98.8%	98.7%	98.8%	99.7%	100.0%	98.9%	98.8%	**S**
Set 2/STRONG	1S1W1	84.9%	85.2%	77.2%	76.3%	78.4%	77.8%	66.7%	66.9%	18.8%	18.5%	**M**
	1S250W1	88.1%	88.8%	86.0%	85.5%	79.9%	80.7%	76.4%	77.6%	27.6%	27.1%	**M**
	1S300W1	87.3%	88.3%	85.9%	85.7%	79.5%	79.4%	78.3%	78.8%	32.7%	31.4%	**M**
	1S350W1	85.0%	85.2%	81.7%	82.4%	78.1%	78.2%	70.8%	70.6%	16.7%	17.9%	**M**
	1S400W1	88.6%	88.1%	86.2%	85.7%	79.0%	79.8%	72.2%	73.2%	21.0%	19.6%	**M**
	1S450W1	86.3%	85.8%	83.9%	85.5%	78.4%	79.5%	69.9%	68.9%	19.4%	18.7%	**M**
	1S500W1	87.8%	87.6%	85.5%	85.4%	79.6%	80.8%	71.8%	71.6%	21.9%	22.2%	**M**
	1S600W1	87.3%	87.5%	85.0%	84.3%	80.3%	80.6%	73.0%	73.1%	21.1%	20.6%	**M**
	1S700W1	84.9%	85.0%	82.6%	82.7%	77.3%	77.6%	70.2%	70.3%	19.9%	20.2%	**M**
	1S800W1	86.5%	86.3%	80.8%	81.5%	79.4%	80.5%	69.0%	69.6%	14.0%	13.1%	**M**
	1S900W1	82.9%	83.0%	79.3%	80.8%	75.3%	76.7%	66.4%	66.9%	17.6%	17.4%	**M**
	S1100W1	82.5%	86.7%	80.3%	80.4%	76.1%	75.5%	77.9%	75.6%	44.6%	50.5%	**M**
	S1150W1	82.5%	82.0%	82.5%	82.0%	75.6%	73.0%	74.0%	73.0%	26.3%	27.5%	**M**
	S1200W1	84.5%	85.7%	81.5%	80.7%	76.8%	75.7%	72.1%	70.6%	30.4%	30.1%	**M**
	S1C50W01	67.9%	74.4%	62.4%	51.4%	60.3%	51.4%	62.9%	70.5%	29.8%	26.3%	**M**

Table 6. Results for KNP instances (objective 2).

	Problem	Multi-Objective						Single-Objective				Test
		NSGAII		MOEA/D		SMSEMOA		GA		ES		
		Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	
Set 1A	2KP100-50	76.4%	77.2%	86.6%	87.4%	79.5%	80.4%	97.0%	97.1%	95.1%	95.5%	**S**
	2KP50-11	81.8%	84.7%	82.2%	86.4%	88.6%	90.4%	98.5%	100.0%	93.7%	100.0%	**S**
	2KP50-50	85.9%	86.6%	90.9%	90.9%	87.3%	88.3%	97.3%	97.7%	94.1%	93.9%	**S**
	2KP50-92	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	99.6%	100.0%	98.1%	100.0%	**M**
	2KP500-41	63.4%	64.2%	57.1%	57.2%	61.3%	61.9%	97.2%	97.3%	91.6%	91.9%	**S**
Set 1B	2KP100-1A	74.6%	74.9%	84.2%	84.8%	75.3%	75.8%	97.1%	97.4%	92.5%	93.6%	**S**
	2KP100-1B	74.3%	75.1%	83.4%	83.4%	77.0%	77.7%	97.3%	97.7%	93.1%	93.5%	**S**
	2KP100-1C	82.2%	82.3%	86.9%	86.5%	82.7%	82.1%	97.4%	97.9%	94.1%	94.3%	**S**
	2KP100-1D	76.4%	76.5%	81.3%	81.5%	76.8%	77.0%	96.4%	96.4%	92.0%	92.3%	**S**
	2KP200-1A	69.9%	70.3%	77.1%	76.5%	70.0%	70.5%	96.4%	96.5%	91.7%	91.6%	**S**
	2KP200-1B	69.8%	70.5%	75.9%	76.3%	70.4%	70.6%	96.7%	97.0%	91.0%	91.2%	**S**
	2KP200-1C	60.9%	61.1%	65.3%	65.4%	63.3%	63.4%	97.4%	97.4%	89.5%	89.8%	**S**
	2KP200-1D	63.6%	63.8%	71.4%	71.0%	64.1%	63.8%	96.8%	96.9%	90.3%	90.4%	**S**
	2KP300-1A	67.1%	67.2%	71.2%	71.5%	66.8%	66.7%	95.6%	95.6%	89.0%	88.5%	**S**
	2KP300-1B	68.5%	68.8%	73.0%	73.1%	68.3%	68.6%	96.6%	96.5%	90.8%	90.9%	**S**
	2KP300-1C	67.1%	67.3%	64.5%	64.4%	66.8%	66.0%	96.4%	96.4%	87.1%	87.0%	**S**
	2KP300-1D	82.9%	83.1%	84.7%	84.8%	80.7%	80.5%	96.7%	97.0%	92.2%	92.6%	**S**
	2KP400-1A	66.2%	66.7%	73.3%	73.4%	66.1%	66.1%	94.8%	94.8%	88.6%	88.5%	**S**
	2KP400-1B	67.6%	67.9%	73.7%	73.8%	66.7%	67.4%	94.6%	94.6%	89.3%	89.8%	**S**
	2KP400-1C	69.2%	69.3%	73.6%	73.6%	67.8%	68.1%	96.2%	96.1%	88.9%	88.9%	**S**
	2KP400-1D	54.4%	54.2%	58.3%	58.3%	54.4%	54.1%	97.3%	97.3%	86.7%	86.4%	**S**
	2KP500-1A	67.0%	66.8%	70.7%	70.6%	66.3%	66.2%	97.2%	97.6%	88.5%	88.6%	**S**
	2KP500-1B	65.1%	65.6%	69.2%	68.7%	64.5%	64.0%	95.0%	94.9%	86.9%	87.0%	**S**
	2KP500-1C	72.1%	72.1%	70.0%	70.7%	70.1%	70.7%	96.1%	96.0%	86.2%	85.6%	**S**
	2KP500-1D	71.7%	71.9%	72.5%	72.4%	70.0%	69.9%	94.0%	94.3%	87.3%	87.2%	**S**
set 2/UNCOR	F5050W01	86.9%	87.8%	93.4%	93.5%	82.7%	83.4%	97.2%	98.0%	95.3%	95.9%	**S**
	F5050W02	89.1%	89.5%	93.7%	94.0%	85.1%	86.2%	97.7%	98.9%	93.1%	92.8%	**S**
	F5050W03	88.6%	89.1%	95.0%	94.7%	86.6%	87.2%	98.1%	100.0%	95.8%	95.1%	**S**
	F5050W04	84.7%	85.2%	89.9%	90.5%	82.8%	83.9%	94.6%	94.5%	91.8%	91.9%	**S**
	F5050W05	91.8%	91.7%	96.2%	98.7%	84.1%	86.7%	98.7%	99.0%	97.4%	98.7%	**S**
	F5050W06	88.0%	88.6%	90.8%	91.2%	86.5%	86.7%	98.4%	100.0%	93.8%	94.4%	**S**
	F5050W07	91.4%	91.7%	98.6%	99.9%	89.6%	91.3%	99.7%	100.0%	99.3%	99.9%	**S**
	F5050W08	93.4%	94.7%	96.8%	97.4%	91.9%	93.9%	98.6%	98.9%	97.1%	97.7%	**S**
	F5050W09	94.9%	95.4%	98.2%	98.6%	93.0%	94.3%	98.4%	99.5%	97.0%	97.1%	-
	F5050W10	90.0%	90.6%	92.4%	91.0%	88.1%	89.8%	98.0%	99.3%	94.3%	93.1%	**S**
	K5050W01	87.0%	86.0%	92.9%	93.4%	84.7%	85.8%	97.3%	97.1%	95.0%	96.1%	**S**
	K5050W02	91.0%	89.3%	95.3%	100.0%	87.3%	87.2%	98.1%	100.0%	96.1%	96.0%	**S**
	K5050W03	89.9%	91.0%	92.6%	91.1%	86.3%	86.7%	98.1%	99.6%	93.0%	92.9%	**S**
	K5050W04	85.7%	87.2%	93.7%	93.8%	83.3%	84.7%	98.7%	98.9%	96.8%	99.3%	**S**
	K5050W05	87.0%	87.3%	93.7%	94.7%	84.5%	84.4%	98.8%	99.2%	97.2%	98.1%	**S**
	K5050W06	90.5%	91.3%	94.2%	93.5%	88.1%	88.5%	97.1%	99.8%	91.8%	92.1%	**S**
	K5050W07	84.2%	85.6%	91.9%	92.3%	81.4%	82.9%	97.7%	98.5%	95.6%	94.6%	**S**

Table 6. *Cont.*

	Problem	Multi-Objective						Single-Objective				Test
		NSGAII		MOEA/D		SMSEMOA		GA		ES		
		Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	
	K5050W08	94.1%	94.3%	96.8%	97.0%	92.3%	93.5%	97.8%	97.8%	95.3%	96.1%	S
	K5050W09	90.2%	91.7%	95.2%	98.7%	85.1%	86.7%	99.6%	100.0%	97.7%	99.9%	S
	K5050W10	81.1%	80.8%	87.4%	87.5%	78.2%	79.0%	97.5%	97.5%	95.1%	96.1%	S
set 2/WEAK	4W150W1	95.5%	95.6%	95.7%	96.0%	94.9%	95.2%	96.1%	96.6%	93.4%	93.5%	-
	4W1W1	86.5%	86.4%	93.1%	93.5%	89.6%	90.0%	94.5%	94.7%	85.7%	85.9%	S
	4W200W1	93.6%	93.9%	93.4%	93.8%	93.3%	93.6%	93.8%	94.2%	89.9%	89.9%	-
	4W250W1	91.9%	91.9%	92.4%	92.3%	90.6%	90.7%	93.2%	93.3%	89.2%	89.6%	S
	4W300W1	93.6%	93.5%	94.6%	94.4%	92.1%	92.1%	95.3%	95.4%	91.4%	91.2%	-
	4W350W1	93.3%	93.7%	94.4%	94.4%	92.7%	92.4%	96.2%	96.5%	90.9%	90.9%	S
	4W400W1	90.8%	91.4%	92.5%	92.4%	90.7%	90.3%	94.0%	94.3%	88.4%	88.4%	S
	4W450W1	89.8%	89.9%	92.0%	92.0%	90.0%	90.3%	94.2%	94.2%	87.2%	87.5%	S
	4W500W1	90.9%	90.5%	93.2%	93.9%	91.1%	90.6%	95.9%	96.1%	88.4%	88.8%	S
	4W600W1	89.0%	89.2%	92.6%	92.5%	90.1%	90.1%	95.2%	95.2%	87.3%	87.5%	S
	4W700W1	88.3%	88.6%	92.1%	92.2%	90.5%	90.3%	95.0%	95.4%	86.2%	86.0%	S
	4W800W1	87.8%	87.8%	92.8%	93.1%	90.2%	90.1%	95.7%	95.8%	86.6%	86.9%	S
	4W900W1	83.0%	83.2%	89.2%	89.1%	86.5%	86.5%	91.7%	91.7%	82.8%	83.3%	S
	W4100W1	95.5%	96.1%	95.7%	96.2%	94.9%	95.1%	95.7%	96.0%	93.8%	94.0%	-
	W4C50W01	98.6%	98.3%	98.7%	98.3%	98.7%	100.0%	99.2%	100.0%	98.5%	100.0%	S
set 2/STRONG	1S1W1	48.4%	48.4%	30.6%	29.8%	41.0%	41.4%	96.4%	96.5%	86.6%	86.8%	S
	1S250W1	68.4%	68.7%	52.0%	52.4%	54.2%	55.2%	95.1%	95.1%	86.9%	87.0%	S
	1S300W1	65.1%	66.7%	52.8%	52.2%	49.6%	50.5%	94.6%	94.9%	86.3%	86.6%	S
	1S350W1	62.1%	63.4%	47.2%	47.1%	47.3%	47.5%	95.1%	95.1%	87.0%	87.0%	S
	1S400W1	66.9%	66.7%	56.6%	57.5%	54.4%	54.4%	94.5%	94.6%	87.2%	87.1%	S
	1S450W1	61.3%	61.2%	48.8%	49.0%	47.8%	47.9%	95.2%	95.3%	86.6%	86.3%	S
	1S500W1	54.0%	54.2%	39.6%	38.9%	41.2%	41.3%	94.2%	94.0%	84.7%	84.9%	S
	1S600W1	54.9%	55.6%	40.1%	40.7%	42.6%	43.1%	95.0%	95.1%	85.4%	85.1%	S
	1S700W1	58.7%	59.4%	48.2%	47.8%	50.7%	51.6%	95.9%	95.9%	87.4%	87.5%	S
	1S800W1	53.0%	52.5%	38.1%	36.6%	44.6%	44.7%	95.9%	96.1%	86.4%	86.1%	S
	1S900W1	50.8%	50.9%	35.5%	34.2%	40.7%	40.8%	95.4%	95.5%	85.4%	85.7%	S
	S1100W1	76.0%	76.3%	68.3%	68.5%	64.5%	66.5%	94.2%	94.5%	85.9%	86.5%	S
	S1150W1	78.3%	79.7%	70.7%	72.7%	65.6%	67.0%	95.8%	96.1%	91.2%	92.0%	S
	S1200W1	67.7%	68.5%	56.9%	59.5%	51.7%	52.1%	92.5%	92.6%	85.9%	86.4%	S
	S1C50W01	84.8%	87.3%	81.6%	81.8%	79.6%	82.6%	92.5%	92.6%	87.9%	90.9%	S

In the case of the KNP, we see that, in most test cases, the SOEAs obtain the best results, especially gGA, for both objective functions (see Tables 5 and 6). In fact, for those cases, gGA is statistically superior to the corresponding multi-objective algorithms. However, the results of the best-performing MOEAs are very close to those obtained by the best-performing SOEAs. Particularly, we should note the behavior of NSGA-II when optimizing objective 1 of the strongly correlated instances Set2/STRONG (see Table 5). NSGA-II not only provides the best solutions, but it is also statistically superior to gGA in all instances belonging to that group. For those instances, NSGA-II is followed by MOEA/D and SMS-EMOA in the ranking.

With regard to objective 1, Figure 5 shows more information on this ranking, and also that gGA is close to the SMS-EMOA but never exceeds it, ranking fourth. We see that eES is ranked last, well behind the remaining algorithms. In general, for objective 1, the SOEAs are statistically superior in 79% of the instances, the MOEAs in 19%, while in 2% of the instances, the algorithms did not exhibit statistically significant differences between the two approaches. Table 5 also shows that MOEA/D ranks second, with 38%, surpassing the eES in these cases. Table 6 shows the KNP results for objective 2, where we can see that gGA again yielded the best results in 93% of the instances, 1% for MOEAs, while 6% present no statistically significant differences.

In this case, eES swapped the second position in the ranking with MOEA/D (for the Set2/UNCOR and Set2/WEAK instances), where MOEA/D ranks second in 27% of the cases. As a result, we can conclude that, when dealing with strongly correlated instances of the KNP, NSGA-II provides the best results, and in fact has to be executed only once, rather than the multiple executions required with a single-objective approach, like gGA, which would have to be executed twice, one run per objective function being optimized. The above would result in significant savings in terms of the computational resources required to solve this type of instance.

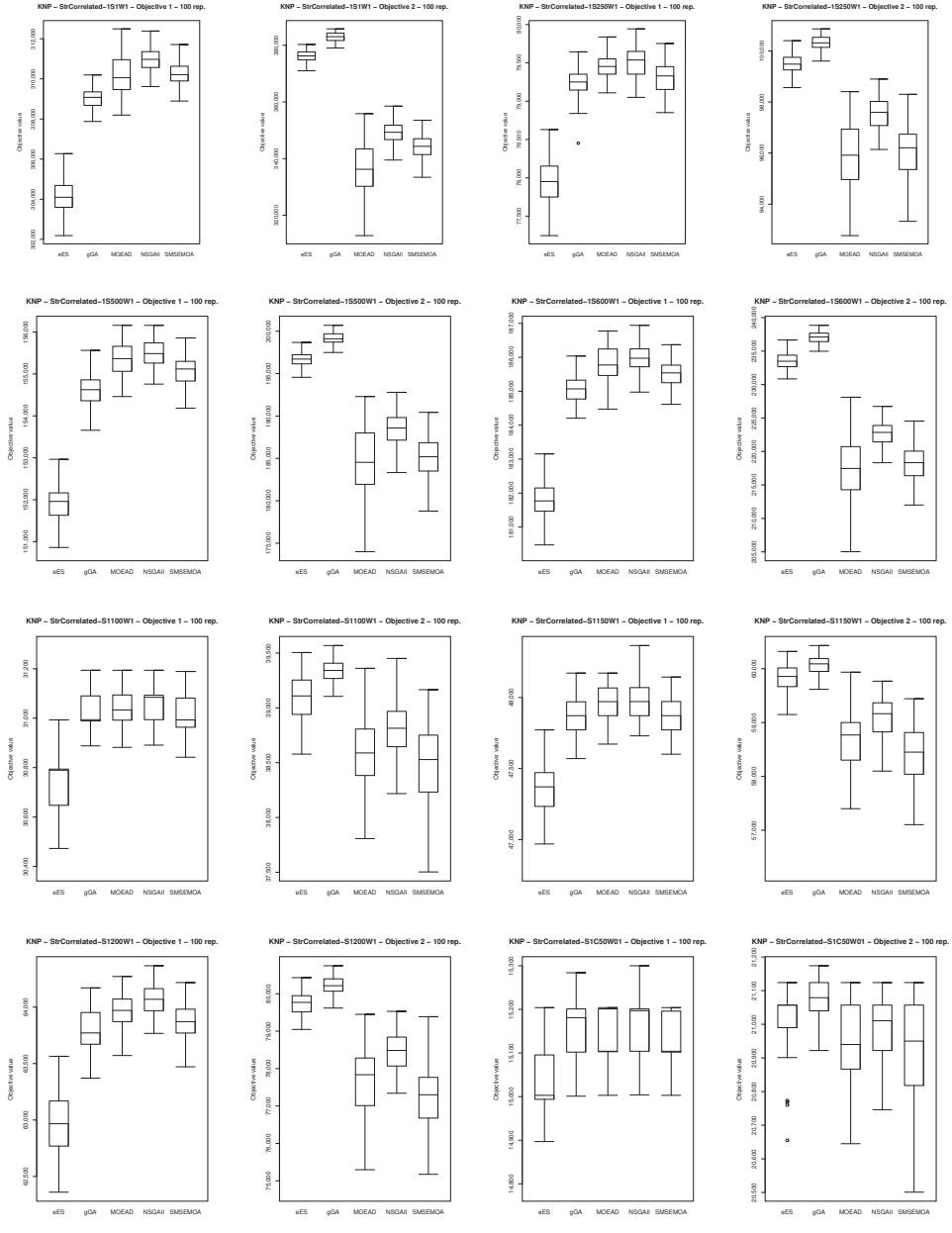


Figure 5. Boxplots showing the results for the KNP (strongly correlated instances) achieved by the different single-objective and multi-objective approaches at the end of 100 repetitions of the runs. Some instances were omitted because of space restrictions. However, all graphics can be found in the repository associated with this paper.

Table 7. Results for TSP instances (objective 1).

Problem	Multi-Objective						Single-Objective				Test
	NSGA-II		MOEA/D		SMS-EMOA		gGA		eES		
	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	
clusAB100	71.3%	72.0%	59.7%	60.3%	22.5%	20.8%	65.6%	67.3%	62.2%	61.2%	M
clusAB300	64.3%	64.5%	56.4%	55.7%	31.2%	32.0%	64.8%	66.2%	63.0%	63.9%	-
clusAB500	62.1%	62.5%	66.2%	68.5%	71.2%	72.9%	31.7%	31.2%	29.5%	30.3%	M
euclAB100	83.2%	82.5%	77.2%	77.2%	29.2%	29.3%	82.6%	82.8%	81.8%	82.45%	-
euclAB300	72.2%	72.3%	68.8%	68.8%	29.8%	30.8%	79.2%	79.2%	78.3%	78.6%	S
euclAB500	72.7%	72.7%	74.0%	73.2%	61.9%	63.3%	42.5%	43.8%	45.7%	45.7%	M
kroAB100	83.1%	83.9%	71.2%	71.1%	27.3%	26.5%	79.0%	78.3%	77.7%	79.1%	M
kroAB1000	69.3%	69.4%	74.3%	74.7%	93.0%	92.7%	11.8%	11.7%	10.9%	10.9%	M
kroAB150	72.7%	73.1%	61.9%	61.6%	20.7%	20.5%	79.1%	80.2%	77.7%	78.2%	S
kroAB200	67.3%	68.8%	58.5%	57.7%	21.9%	22.1%	77.9%	77.2%	78.6%	79.4%	S
kroAB300	69.9%	70.3%	65.2%	64.9%	30.4%	28.6%	76.8%	77.5%	77.6%	79.0%	S
kroAB400	73.7%	73.5%	64.4%	65.0%	39.1%	39.6%	58.2%	57.4%	59.2%	58.4%	M
kroAB500	68.8%	68.4%	68.8%	67.8%	62.8%	63.5%	38.0%	37.8%	38.1%	39.2%	M
kroAB750	67.2%	67.4%	71.7%	71.7%	84.7%	85.4%	19.9%	19.9%	18.8%	18.3%	M
kroAC100	82.6%	83.1%	74.8%	75.4%	31.6%	31.6%	80.1%	79.4%	78.9%	80.2%	M
kroAD100	80.9%	81.0%	72.6%	73.1%	27.4%	25.8%	77.0%	76.4%	75.8%	77.1%	M
kroBC100	81.1%	81.3%	74.2%	73.9%	31.6%	32.3%	77.9%	77.2%	77.1%	76.1%	M
kroBD100	80.8%	81.1%	71.7%	72.9%	26.9%	27.1%	76.5%	75.7%	75.6%	74.6%	M
kroCD100	79.4%	79.9%	70.1%	71.8%	28.0%	29.0%	74.5%	73.9%	76.3%	76.9%	M

Table 8. Results for TSP instances (objective 2).

Problem	Multi-Objective						Single-Objective				Test
	NSGA-II		MOEA/D		SMS-EMOA		gGA		eES		
	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	
clusAB100	78.3%	79.0%	69.7%	69.3%	31.6%	33.5%	70.9%	71.1%	70.1%	71.9%	M
clusAB300	61.7%	61.5%	54.4%	55.4%	27.0%	24.4%	63.9%	64.9%	61.6%	62.4%	-
clusAB500	68.2%	68.8%	69.4%	69.6%	72.8%	73.6%	38.5%	37.7%	38.2%	38.4%	M
euclAB100	84.3%	84.0%	76.4%	75.6%	28.3%	29.7%	81.4%	81.5%	80.0%	81.3%	M
euclAB300	71.2%	72.3%	63.1%	62.5%	23.0%	23.6%	79.5%	80.2%	76.1%	76.7%	S
euclAB500	67.3%	67.1%	67.7%	67.1%	58.1%	58.2%	32.9%	33.2%	33.1%	33.3%	M
kroAB100	81.6%	82.3%	72.0%	73.6%	24.5%	23.9%	77.2%	76.5%	76.3%	75.3%	M
kroAB1000	67.4%	67.2%	72.6%	72.0%	91.7%	91.4%	11.1%	11.2%	10.0%	10.0%	M
kroAB150	74.7%	74.8%	66.5%	66.2%	24.5%	25.0%	78.5%	79.4%	77.4%	78.7%	S
kroAB200	72.2%	72.2%	63.9%	64.0%	24.1%	24.6%	81.8%	82.0%	80.9%	81.2%	S
kroAB300	71.5%	71.7%	65.3%	65.5%	31.6%	31.9%	75.1%	74.2%	76.0%	77.0%	S
kroAB400	70.2%	69.5%	66.9%	67.7%	43.0%	41.5%	62.3%	61.0%	60.7%	60.1%	M
kroAB500	65.2%	66.1%	64.1%	64.1%	54.6%	53.0%	30.9%	33.1%	28.8%	29.8%	M
kroAB750	65.9%	66.2%	70.6%	70.8%	85.3%	85.1%	16.8%	16.9%	16.6%	16.3%	M
kroAC100	76.6%	77.0%	65.5%	66.0%	21.2%	20.0%	71.1%	70.6%	73.0%	73.6%	M
kroAD100	81.2%	81.6%	69.3%	70.0%	21.8%	21.7%	76.4%	76.2%	72.3%	73.5%	M
kroBC100	83.0%	83.0%	74.4%	75.0%	29.6%	29.0%	77.3%	76.7%	79.0%	79.6%	M
kroBD100	79.1%	79.6%	72.3%	74.3%	20.0%	19.7%	77.4%	77.1%	72.9%	74.2%	-
kroCD100	82.3%	82.8%	72.6%	73.4%	28.7%	28.9%	75.3%	74.7%	77.1%	77.7%	M

Regarding the TSP, the results for objective 1 (Table 7) and objective 2 (Table 8), show hardly any differences. In both cases, NSGA-II was the best-performing approach, not only considering almost all small instances, but also some large ones. Furthermore, in those cases where NSGA-II was superior, the differences were statistically significant compared the corresponding best-performing single-objective approach. As in the case of the strongly correlated instances of the KNP, for those particular instances of the TSP, it is better to run a multi-objective approach, such as NSGA-II, instead of running a single-objective algorithm. As a first approach, decision makers usually tend to perform a transformation of a multi-objective problem into a single-objective one, in the case they are interested in a particular objective of a multi-objective problem. The said transformation is carried out either by performing a scalarization of the different objective functions or by redefining objective functions as constraints. Bearing the above in mind, although practitioners are only focused on one of the objective functions of a multi-objective problem, the quality of the solutions attained by the direct application of a multi-objective optimizer could be higher in comparison to the quality of the solutions achieved by a single-objective algorithm executed for each of the objective functions independently. As a result, from the practical point of view, the application of a multi-objective solver

to a multi-objective problem could be a much better option rather than performing a transformation of the multi-objective problem into a single-objective one to solve it through a single-objective approach.

Finally, we note that for most instances with a size between 150 and 300, MOEAs are dominated by SOEAs. In larger instances, the SMS-EMOA tends to be superior to the other approaches. In general, and considering both objective functions, the MOEAs are statistically superior in 69% of the instances, SOEAs in 21%, while 10% exhibit no statistically significant differences, with the NSGA-II being the best-ranked algorithm, followed by gGA, and finally by SMS-EMOA, eES, and MOEA/D. Moreover, if we consider how MOEAs behave with the TSP problem, we see that for problem instances with sizes of 100, 300, 500, 750 and 1000 (see Figures 3 and 4, Tables 6 and 7), MOEAs—especially SMS-EMOA—can perform better than SOEAs as the size of the instances increases. Figure 6 provides more statistical information. For example, note the significant difference in the behavior of SMS-EMOA between small and large instances.

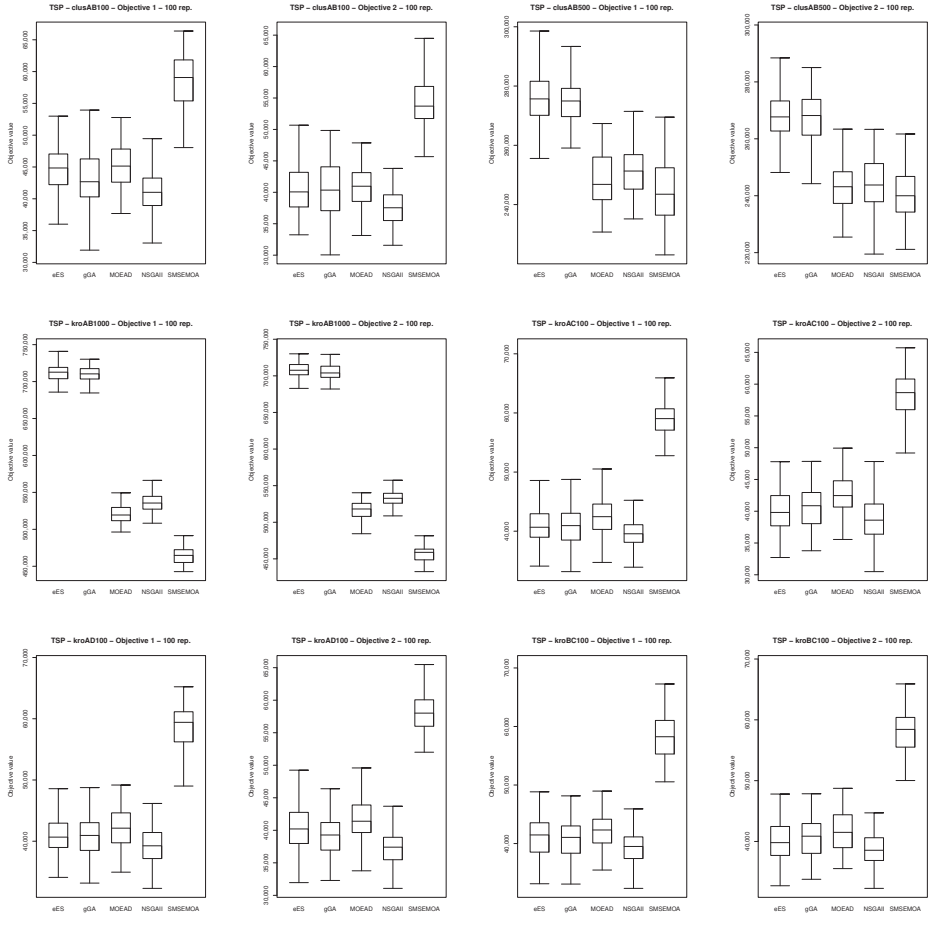


Figure 6. *Cont.*

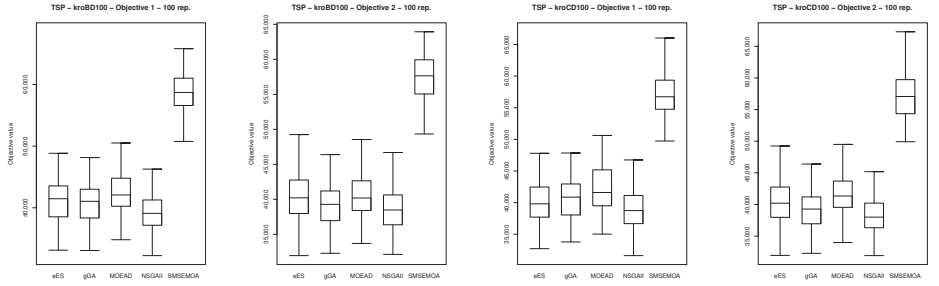


Figure 6. Boxplots showing the results for the TSP achieved by the different single-objective and multi-objective approaches at the end of 100 repetitions of the runs. Some instances were omitted because of space restrictions. However, all graphics can be found in the repository associated with this paper.

5. Conclusions

In this work, we have studied the assessment of multi-objective optimization approaches when trying to optimize single-objective problems. From our point of view, it is interesting to analyze the differences between the solutions provided by these multi-objective techniques (when considering each objective value separately) and those reached by algorithms that are specifically designed to optimize single and independent objectives. For this reason, in this paper, we presented a comparative study between Multi-Objective Evolutionary Algorithms and Single-Objective Evolutionary Algorithms. For the experimental analysis, we focused on two well-known and widely studied optimization problems: the Knapsack Problem and the Travelling Salesman Problem. We considered bi-objective formulations of the aforementioned problems. These bi-objective optimization problems were directly—in a single run—processed using the multi-objective approaches, thus yielding a Pareto front, from which we only are interested in two values: the point optimizing objective 1 and, separately, the point optimizing objective 2. Meanwhile, the single-objective approaches must be executed twice: once to optimize objective 1, defined in the bi-objective formulation of the problem, and again to optimize objective 2.

The computational study carried out allows us to conclude that although MOEAs have to deal with several objectives simultaneously, in some cases they have proven to be more effective than single-objective approaches. In particular, the multi-objective approaches exhibited better behavior when dealing with larger instances or with instances where the objectives are strongly correlated. For those specific cases, the direct application of a multi-objective solver to a multi-objective problem is a better choice in comparison to the transformation of the multi-objective problem into a single-objective one to be solved by means of a single-objective algorithm. This conclusion can be explained by the intrinsic capacity of MOEAs to maintain diversity within a population. MOEAs need conflicting objectives and more time to converge, thus performing a larger exploration of the solution space. The more negatively correlated the objectives, the more they conflict one with each other. Otherwise, if we consider a context with non-conflicting objective functions, the Pareto front converges to a single point. Hence, in these cases, it is better to address the problem by optimizing independently each of the objective functions through a single-objective algorithm.

Considering the above, in the future, further evaluations should be done with a more—representative and independent— set of problems and instances. We could thus further investigate the key factors influencing the improvement of MOEA approaches to single-objective environments. Since the design of MOEAs allows each objective to have a helper-objective effect on the other objective, this property can provide more freedom to maintain the diversity of individuals within a population. Such a feature is not present under the single-objective approaches.

It is important to note that what is sought is useful diversity. A greater diversity does not necessarily imply a proper balance between exploration and exploitation, so a high diversity might be counterproductive. In this work, we did not employ a suitable diversity management strategy because our intention was to study the intrinsic capacity of MOEAs to maintain diversity and to analyze how effective these approaches are in single-objective optimization. However, after this initial analysis, it would be worthwhile to design new experiments were the intrinsic and specific features of MOEAs could be evaluated separately in some way.

Author Contributions: Conceptualization, G.M., C.L. and E.S.; Formal analysis, M.M., G.M., C.L. and E.S.; Funding acquisition, M.M. and C.L.; Investigation, M.M., G.M., C.L. and E.S.; Methodology, G.M. and C.L.; Software, M.M., G.M. and E.S.; Supervision, G.M., C.L. and E.S.; Validation, G.M., C.L. and E.S.; Visualization, M.M., G.M. and E.S.; Writing of original draft, M.M., G.M., C.L. and E.S.; Writing of review & editing, M.M., G.M., C.L. and E.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially funded by the Spanish Ministry of Economy, Industry and Competitiveness as part of the program "I+D+i Orientada a los Retos de la Sociedad" [contract number TIN2016-78410-R]. The work of Mohammed Mahrach was funded by the Canary Government "Agencia Canaria de Investigación Innovación y Sociedad de la Información—ACIISI" [contract number TESIS2018010095].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Eiben, A.E.; Smith, J.E. *Introduction to Evolutionary Computing*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2015.
2. Ma, H.; Shen, S.; Yu, M.; Yang, Z.; Fei, M.; Zhou, H. Multi-population techniques in nature inspired optimization algorithms: A comprehensive survey. *Swarm Evol. Comput.* **2019**, *44*, 365–387. [CrossRef]
3. Sloss, A.N.; Gustafson, S. 2019 Evolutionary Algorithms Review. In *Genetic Programming Theory and Practice XVII*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 307–344.
4. Segura, C.; Coello, C.A.C.; Miranda, G.; León, C. Using multi-objective evolutionary algorithms for single-objective constrained and unconstrained optimization. *Ann. Oper. Res.* **2016**, *240*, 217–250. [CrossRef]
5. Coello, C.A.C. *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd ed.; Genetic Algorithms and Evolutionary Computation; Springer: New York, NY, USA, 2007.
6. Deb, K.; Goldberg, D.E. An Investigation of Niche and Species Formation in Genetic Function Optimization. In Proceedings of the 3rd International Conference on Genetic Algorithms, Fairfax, VA, USA, 4–7 June 1989; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1989; pp. 42–50.
7. Pulido, G.T.; Coello, C.A.C. Using Clustering Techniques to Improve the Performance of a Multi-objective Particle Swarm Optimizer. In Proceedings of the Genetic and Evolutionary Computation—GECCO 2004, Seattle, WA, USA, 26–30 June 2004; Deb, K., Ed.; Springer: Berlin/Heidelberg, Germany, 2004; pp. 225–237.
8. Wang, Y.N.; Wu, L.H.; Yuan, X.F. Multi-Objective Self-Adaptive Differential Evolution with Elitist Archive and Crowding Entropy-Based Diversity Measure. *Soft Comput.* **2010**, *14*, 193–209. [CrossRef]
9. Watanabe, S.; Sakakibara, K. Multi-objective approaches in a single-objective optimization environment. In Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2–5 September 2005; Volume 2, pp. 1714–1721.
10. De Armas, J.; León, C.; Miranda, G.; Segura, C. Optimisation of a multi-objective two-dimensional strip packing problem based on evolutionary algorithms. *Int. J. Prod. Res.* **2010**, *48*, 2011–2028. [CrossRef]
11. Segura, C.; Coello, C.A.C.; Segredo, E.; Miranda, G.; León, C. Improving the diversity preservation of multi-objective approaches used for single-objective optimization. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; pp. 3198–3205.
12. Knowles, J.D.; Watson, R.A.; Corne, D.W. Reducing Local Optima in Single-Objective Problems by Multi-objectivization. In *Evolutionary Multi-Criterion Optimization*; Zitzler, E., Thiele, L., Deb, K., Coello, C.A.C., Corne, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2001; pp. 269–283.
13. Mezura-Montes, E.; Coello, C.A.C. Constrained Optimization via Multiobjective Evolutionary Algorithms. In *Multiobjective Problem Solving from Nature: From Concepts to Applications*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 53–75.

14. Bui, L.; Abbass, H.; Branke, J. Multiobjective optimization for dynamic environments. In Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2–5 September 2005; Volume 3, pp. 2349–2356.
15. Jensen, M.T. Helper-Objectives: Using Multi-Objective Evolutionary Algorithms for Single-Objective Optimisation. *J. Math. Model. Algorithms* **2004**, *3*, 323–347. [CrossRef]
16. Martello, S.; Toth, P. *Knapsack Problems: Algorithms and Computer Implementations*; John Wiley & Sons: Hoboken, NJ, USA, 1990.
17. Shmoys, D.; Lenstra, J.; Kan, A.; Lawler, E. *The Traveling Salesman Problem*; A Wiley-Interscience Publication; John Wiley & Sons: Hoboken, NJ, USA, 1985.
18. Bengio, Y.; Lodi, A.; Prouvost, A. Machine learning for combinatorial optimization: A methodological tour d'horizon. *Eur. J. Oper. Res.* **2020**. [CrossRef]
19. Lombardi, M.; Milano, M. Boosting Combinatorial Problem Modeling with Machine Learning. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, Stockholm, Sweden, 13–19 July 2018; pp. 5472–5478.
20. Vamvakas, P.; Tsiropoulou, E.E.; Papavassiliou, S. Dynamic Provider Selection & Power Resource Management in Competitive Wireless Communication Markets. *Mob. Netw. Appl.* **2018**, *23*, 86–99.
21. Captivo, M.E.; Clìmaco, J.A.; Figueira, J.; Martins, E.; Santos, J.L. Solving Bicriteria 0-1 Knapsack Problems Using a Labeling Algorithm. *Comput. Oper. Res.* **2003**, *30*, 1865–1886. [CrossRef]
22. Gandibleux, X. MOCOlib: Multi-Objective Combinatorial Optimization Library. Available online: http://xgandibleux.free.fr/MOCOlib/MOKP.html (accessed on 11 November 2020).
23. Gandibleux, X.; Freville, A. Tabu Search Based Procedure for Solving the 0-1 MultiObjective Knapsack Problem: The Two Objectives Case. *J. Heuristics* **2000**, *6*, 361–383. [CrossRef]
24. Visée, M.; Teghem, J.; Pirlot, M.; Ulungu, E. Two-phases Method and Branch and Bound Procedures to Solve the Bi–objective Knapsack Problem. *J. Glob. Optim.* **1998**, *12*, 139–155. [CrossRef]
25. Ehrgott, M.; Gandibleux, X. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum* **2000**, *22*, 425–460. [CrossRef]
26. Robinson, J. *On the Hamiltonian Game (A Traveling Salesman Problem)*; Technical Report; Rand Project Air Force: Arlington, VA, USA, 1949.
27. Eiselt, H.A.; Sandblom, C.L. Traveling Salesman Problems and Extensions. In *Integer Programming and Network Models*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 315–341.
28. Florios, K.; Mavrotas, G. Generation of the exact Pareto set in Multi-Objective Traveling Salesman and Set Covering Problems. *Appl. Math. Comput.* **2014**, *237*, 1–19. [CrossRef]
29. Lust, T. Speed-up techniques for solving large-scale bTSP with the two-phase pareto local search. In Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, Atlanta, GA, USA, 12–16 July 2008; pp. 761–762.
30. Paquete, L.; Stutzle, T. Design and analysis of stochastic local search for the multiobjective traveling salesman problem. *Comput. Oper. Res.* **2009**, *36*, 2619–2631. [CrossRef]
31. Reinelt, G. TSPLIB—A Traveling Salesman Problem Library. *ORSA J. Comput.* **1991**, *3*, 376–384. [CrossRef]
32. Reinelt, G. TSPLIB: Library of Traveling Salesman Problems. Available online: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ (accessed on 11 November 2020).
33. Lust, T.; Teghem, J. Two-phase Pareto local search for the biobjective traveling salesman problem. *J. Heuristics* **2010**, *16*, 475–510. [CrossRef]
34. Cui, Z.; Gao, X.Z. Special issue on evolutionary multi-objective optimization (EMO): Theory and applications. *Int. J. Mach. Learn. Cybern.* **2019**, *10*, 1927–1929. [CrossRef]
35. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
36. Zitzler, E.; Laumanns, M.; Thiele, L. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*; Springer: Berlin, Germany, 2001; pp. 95–100.
37. Corne, D.W.; Jerram, N.R.; Knowles, J.D.; Oates, M.J.; J, M. PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001), San Francisco, CA, USA, 7–11 July 2001; pp. 283–290.

38. Ishibuchi, H.; Murata, T. Multi-objective genetic local search algorithm (MOGLS). In Proceedings of the IEEE International Conference on Evolutionary Computation, Nagoya, Japan, 20–22 May 1996; pp. 119–124.
39. Murata, T.; Ishibuchi, H.; Gen, M. Cellular Genetic Local Search for Multi-Objective Optimization. In Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, CA, USA, 7–11 July 2000; pp. 307–314.
40. Zhang, Q.; Li, H. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Trans. Evol. Comput.* **2007**, *11*, 712–731. [CrossRef]
41. Zitzler, E.; Künzli, S. Indicator-Based Selection in Multiobjective Search. In Proceedings of the Conference on Parallel Problem Solving from Nature (PPSN VIII), Birmingham, UK, 18–22 September 2004; pp. 832–842.
42. Beume, N.; Naujoks, B.; Emmerich, M. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *Eur. J. Oper. Res.* **2007**, *181*, 1653–1669. [CrossRef]
43. Jiang, S.; Zhang, J.; Ong, Y.; Zhang, A.N.; Tan, P.S. A Simple and Fast Hypervolume Indicator-Based Multiobjective Evolutionary Algorithm. *IEEE Trans. Cybern.* **2015**, *45*, 2202–2213. [CrossRef]
44. Hernandez Gomez, R.; Coello, C.A.C. MOMBI: A new metaheuristic for many-objective optimization based on the R2 indicator. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; pp. 2488–2495.
45. Wolpert, D.; Macready, W. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
46. Cobb, H.G.; Grefenstette, J.J. GA for Tracking Changing Environment. In Proceedings of the 5th International Conference on GA, San Francisco, CA, USA, 8–13 August 1993.
47. Whitley D. GENITOR: A different Genetic Algorithm. In Proceedings of the Rocky Mountain Conference on Artificial Intelligence, Boulder, CO, USA, 29 March–1 April 1988.
48. Rechenberg, I. Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution. Ph.D. Thesis, Technische Universität Berlin, Berlin, Germany, 1973.
49. Oliveto, P.S.; He, J.; Yao, X. Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. *Int. J. Autom. Comput.* **2007**, *4*, 281–293. [CrossRef]
50. Droste, S.; Jansen, T.; Wegener, I. On the Analysis of the (1 + 1) Evolutionary Algorithm. *Theor. Comput. Sci.* **2002**, *276*, 51–81. [CrossRef]
51. Curry, D.M.; Dagli, C.H. Computational Complexity Measures for Many-objective Optimization Problems. *Procedia Comput. Sci.* **2014**, *36*, 185–191. [CrossRef]
52. Tian, Y.; Wang, H.; Zhang, X.; Jin, Y. Effectiveness and efficiency of non-dominated sorting for evolutionary multi- and many-objective optimization. *Complex Intell. Syst.* **2017**, *3*, 247–263. [CrossRef]
53. Saxena, D.K.; Duro, J.A.; Tiwari, A.; Deb, K.; Zhang, Q. Objective Reduction in Many-Objective Optimization: Linear and Nonlinear Algorithms. *IEEE Trans. Evol. Comput.* **2013**, *17*, 77–99. [CrossRef]
54. Durillo, J.J.; Nebro, A.J. jMetal: A Java framework for multi-objective optimization. *Adv. Eng. Softw.* **2011**, *42*, 760–771. [CrossRef]
55. López-Ibáñez, M.; Dubois-Lacoste, J.; Pérez Cáceres, L.; Birattari, M.; Stützle, T. The irace package: Iterated racing for automatic algorithm configuration. *Oper. Res. Perspect.* **2016**, *3*, 43–58. [CrossRef]
56. Segura, C.; Coello, C.; Segredo, E.; Aguirre, A.H. A Novel Diversity-Based Replacement Strategy for Evolutionary Algorithms. *IEEE Trans. Cybern.* **2016**, *46*, 3233–3246. [CrossRef]

mathematics

MDPI

Article

MooFuzz: Many-Objective Optimization Seed Schedule for Fuzzer

Xiaoqi Zhao, Haipeng Qu *, Wenjie Lv, Shuo Li and Jianliang Xu

College of Information Science and Engineering, Ocean University of China, Qingdao 266100, China; zhaoxiaoqi@stu.ouc.edu.cn (X.Z.); lwj3656@stu.ouc.edu.cn (W.L.); li_shuo@stu.ouc.edu.cn (S.L.); xjl9898@ouc.edu.cn (J.X.)

* Correspondence: quhaipeng@ouc.edu.cn

Abstract: Coverage-based Greybox Fuzzing (CGF) is a practical and effective solution for finding bugs and vulnerabilities in software. A key challenge of CGF is how to select conducive seeds and allocate accurate energy. To address this problem, we propose a novel many-objective optimization solution, MooFuzz, which can identify different states of the seed pool and continuously gather different information about seeds to guide seed schedule and energy allocation. First, MooFuzz conducts risk marking in dangerous positions of the source code. Second, it can automatically update the collected information, including the path risk, the path frequency, and the mutation information. Next, MooFuzz classifies seed pool into three states and adopts different objectives to select seeds. Finally, we design an energy recovery mechanism to monitor energy usage in the fuzzing process and reduce energy consumption. We implement our fuzzing framework and evaluate it on seven real-world programs. The experimental results show that MooFuzz outperforms other state-of-the-art fuzzers, including AFL, AFLFast, FairFuzz, and PerfFuzz, in terms of path discovery and bug detection.

Keywords: seed schedule; many-objective optimization; fuzzing; bug detection; path discovery

Citation: Zhao, X.; Qu, H.; Lv, W.; Li, S.; Xu, J. MooFuzz: Many-Objective Optimization Seed Schedule for Fuzzer. *Mathematics* **2021**, *9*, 205. https://doi.org/10.3390/math9030205

Academic Editors: Amir H. Alavi and Gai-Ge Wang

Received: 22 December 2020

Accepted: 16 January 2021

Published: 20 January 2021

1. Introduction

Fuzzing is a popular and effective software testing technology for detecting bugs and vulnerabilities. In the past few years, it has gained widespread usage in mainstream software companies (such as Google [1–3], Microsoft [4], and Adobe [5]) and has found thousands of vulnerabilities.

Coverage-based Greybox Fuzzing (CGF) [6,7] is one of the most popular methods of fuzzing. It is based on the guidance that increasing code coverage usually leads to better crash detection. By using lightweight instrumentation, CGF automatically generates a large number of inputs to feed target programs, and continuously collects coverage information as feedback to guide fuzzing.

Inspired by the impressive achievements of CGF, many researchers have conducted studies and developed their own fuzzers from different perspectives [8–10]. AFLFast [11] assigns more energy to the low-frequency paths based on the Markov chain model. AFLGo [12], a directed grey-box fuzzer, is implemented to generate inputs to reach given sets of target program locations. FairFuzz [13] identifies rare branches in the program and adjusts mutation strategies to increase coverage. MOPT [14] leverages a mutation schedule based on particle swarm optimization (PSO) to accelerate the convergence speed. EcoFuzz [15] improves the power schedule for discovering new paths using a variant of the adversarial multi-armed bandit model. PerfFuzz [16] generates pathological inputs to detect algorithm complexity vulnerabilities. MemLock [17] utilizes memory consumption information to guide seed selection to trigger the weakness of memory corruption.

However, most previous approaches mainly leverage a single selection criterion to select seeds. While these approaches are simple and easy to use in solving specific problems,

they are still inadequate to reach effective coverage and detect bugs within a reasonable amount of time. Cerebro [18] uses many objectives as the seed selection criteria, but it cannot dynamically adjust the seed selection strategy according to the fuzzing process. As a result, much useful information is ignored, affecting the discovery of bugs and paths.

In this paper, we propose a many-objective optimization seed schedule model, named MooFuzz, which is aimed at speeding up bug discovery and improving code coverage. MooFuzz performs static analysis on the code and marks the risky locations in order to collect edge risk information. In the fuzzing process, a novel measurement method is used to update useful information including the path risk, the path frequency, and the mutation information. According to the fuzzing process, MooFuzz divides the seed pool state into three categories: Exploration State, Search State, and Assessment State. In Exploration State, the fuzzer emphasizes the exploration of high-risk locations in the program. In Search State, the fuzzer spends more energy to find the new path. In Assessment State, the fuzzer aims to select and evaluate promising seeds. MooFuzz collects different information to measure the priority of seeds in each state and builds a many-objective optimization model to select optimal seed set using a non-dominated sorting algorithm [19]. Beside, we also observe that many studies have improved power schedule method, but they have not performed energy monitoring in power schedule. Therefore, MooFuzz uses multiple information to set the energy for selected seeds and monitors energy usage.

We design and implement our prototype by extending American Fuzzy Lop (AFL) [7], and evaluate it against for popular fuzzers AFL, AFLFast, FairFuzz, and PerfFuzz in terms of path discovery and bug detection. We conduct our evaluation on seven real-world applications. The experimental results demonstrate that MooFuzz performs better than others. Compared with AFL, AFLFast, Fairfuzz, and PerfFuzz, it triggers 46%, 32%, 34%, and 153% more crashes with almost the same execution time, respectively. Furthermore, we run cases and analyze the discovery of vulnerabilities. MooFuzz is able to trigger stack overflow, heap overflow, null pointer dereference, and memory leaks related vulnerabilities.

The contributions of this paper are as follows.

- We propose the path risk measurement method to assist seed schedule in Exploration State.
- We use many-objective optimization to model CGF and classify three different states of seed pool and put forth different selection criteria that enhance the fuzzer performance.
- We propose an energy allocation and monitor mechanism to improve the power schedule.
- We implement our framework as MooFuzz and evaluate its effectiveness on a series of popular real-world applications. MooFuzz substantially outperforms the other fuzzers.

The rest of this paper is organized as follows. Section 2 introduces the background and related work of many-objective optimization and CGF. Section 3 shows the design of MooFuzz. Section 4 presents the evaluation and we get the conclusion in Section 5.

2. Background and Related Work

In this section, we introduce the background of many-objective optimization and CGF and discuss related work.

2.1. Many-Objective Optimization

Multi-objective optimization [20–23] falls into the field of multiple criteria decision-making. It optimizes all goals at the same time to get the optimal solution. Therefore, multi-objective optimization problems (MOPs) get a set of solutions. Generally, a MOP is an optimization problem with two or three objectives. A many-objective optimization problem (MaOP) is an optimization problem [24–27] with four or more objectives. In recent years, many researchers have used multi-objective optimization methods to solve practical problems [28–31], such as scheduling [32,33], planning [34–36], fault diagnosis [37–39], classification [40,41], test-sheet composition [42], object extraction [43], variable reduction [44], and virtual machine placement [45]. Multi-objective evolutionary algorithms

(MOEAs), such as non-dominated sorting GA [46], multi-objective particle swarm optimization (MOPSO) [47–49], NSGA-II [50], NSGA-III [51,52], decomposition-based MOEA [53] and corresponding improved versions [54–56], are the most used solutions.

In many-objective optimization problems, minimization problems simultaneously optimize minimize objectives to obtain the maximum benefit. Within the scope of mathematics, minimization problems are embodied in the minimization of objective functions (that is, to minimize all objective values of objective functions as far as possible). In this paper, we use the minimum optimization model to carry out seed schedule. The definition of minimum optimization problems is given below.

$$\begin{cases} Min\ F(x) = [f_1(x), f_2(x), \cdots, f_m(x)]^T \\ \text{s.t. } m > 3 \\ x \in X \subseteq \mathbb{R}^n \end{cases} \tag{1}$$

where $F(x)$ is the objective vector, $f_i(x)$ is the i-th objective to be minimized, $x = (x_1, \cdots, x_n)$ is a vector of n decision variables, X is an n-dimensional decision space, and m denotes the number of objectives to be optimized.

Definition 1 (Pareto Dominance [57]). *Given any two decision vectors x, y with M objectives for the minimization optimization. $\forall x, y \in X$, if there is $f_m(x) \leq f_m(y)$ for all $m = 1, 2, \cdots, M$ then x dominates y, which is denoted as $x \prec y$.*

Definition 2 (Pareto Optimal [57]). *Assuming that $x^* \in X$, if there is no solution $x \in X$ satisfying $x \prec x^*$, then x^* is the Pareto optimal solution.*

Definition 3 (Pareto Optimal Set [57]). *All the Pareto optimal solutions constitute the Pareto optimal set (PS).*

Definition 4 (Pareto Front [57]). *All the objective vectors of the solutions in Pareto optimal set constitute the Pareto front (PF).*

Figure 1 is a solution distribution under two-dimensional objective space, where all points represent solutions. For a minimal optimization problem, it can be seen that the point A is smaller than the point C under the two-dimensional objective space, that is, there is a dominance relationship between the point A and the point C, and the point C is dominated the point A. For the points A and B in Figure 1, we can see that the point A is greater than the point B on the f_2 axis, but the point A is less than B on the f_1 axis, so there is not a dominance relationship between the point A and the point B.

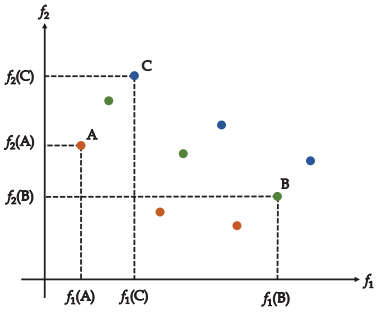


Figure 1. Solutions in a two-dimensional objective space.

2.2. Coverage-Based Greybox Fuzzing

CGF is an evolutionary algorithm that includes two stages: the static analysis stage and the fuzzing loop stage. In the static analysis stage, it executes compile-time or dynamic binary instrumentation to obtain the instrumented target program. In the fuzzing loop stage, CGF uses a series of initial seeds provided by the user as inputs and maintains a seed queue stored in the seed pool. CGF first selects a saved seed input from the seed queue and mutates it to generate the new input by using mutation strategies. Next, the target program is executed with the new input. Then, lightweight instrumentation technique is used to gather coverage information, if the new input causes a crash, it will be marked and added to the crash set. If the new input leads to new coverage, CGF will judge that the new input is interesting and add it to the seed pool. Algorithm 1 shows the workflow of CGF in the fuzzing loop stage.

Algorithm 1: Coverage-based Greybox Fuzzing

Input: a data set of initial *seeds*, an instrumented target program P
Output: a seed queue Q, a crash set C

```
1  Q ← seeds
2  C ← ∅
3  Procedure fuzzing process
4  while TRUE do
5      S ← SeedsSelect(Q)
6      E ← PowerSchedule(S)
7      for mutation in deterministic stage do
8          for i in Range(0, S.length) do
9              S' ← Mutation(S)
10             RunAndSave(P, S')
11     HavocMutation(P, E)
12     RunAndSave(P, S')
13 Procedure RunAndSave(P, S')
14 status ← Run_Target(P, S')
15 if is_NewCoverage(status) then
16     Q ← Q ∪ S'
17     return
18 if is_Crash(status) then
19     C ← C ∪ S'
20     return
```

2.2.1. Code Instrumentation

Code instrumentation aims to insert code fragments at compile-time, which is useful for path tracing and testing during the fuzzing process. AFL [7] is a greybox fuzzer using edge (branch) coverage as feedback. Before the fuzzing loop stage, AFL first uses afl-gcc or afl-clang as instrumentation commands to trace edge coverage. AFL preserves a 64KB shared bitmap *Bitmap* to record edge coverage information including whether the edge has been visited, and the count of hits. AFL assigns a random number to represent each basic block in the program and uses the XOR and right shift operation for the current basic block and the previous basic block to mark each edge. Each edge is used as an offset of *Bitmap* and the value is the count of hits.

The specific formula for coverage calculation is as follows [9].

$$cur\_location = Random() \tag{2}$$

$$Bitmap[cur\_location \oplus prev\_location] + + \tag{3}$$

$$prev\_location = cur\_location >> 1 \quad (4)$$

2.2.2. Seed Schedule

Seed schedule refers to select seeds from the seed pool for future mutation. A perfect seed schedule scheme is conducive to speeding up path discovery and bug detection. AFL [7] gives priority to seeds that are unfuzzed (not selected for mutation) and favored (among all seeds passing through the edge, the seed with the smallest product of seed length and execution time). AFLGo [12] preferentially selects seeds closer to the target location for directed fuzzing. VUzzer [8] prioritizes seeds of deeper paths, it may detect bugs deep in the code. SlowFuzz [58] preferentially selects seeds that generate more resource consumption to trigger algorithm complexity vulnerabilities. In order to discover memory consumption bugs, MemLock [17] preferentially selects seed inputs that generate more memory consumption. UAFL [59] preferentially selects seeds that execute the operation sequence violating typestate properties to uncover use-after-free (UAF) vulnerabilities.

2.2.3. Mutation Strategy

The mutation strategy determines where and how to mutate the selected seed. Different fuzzers use different mutation strategies. AFL has two mutation stages: the deterministic stage and the indeterministic stage.

The deterministic stage. The deterministic stage is used when the first time fuzzing seed. This stage includes mutation operators, bitflip, byteflip, arithmetic addition/subtraction, interesting values, and dictionary.

The indeterministic stage. After completing the deterministic stage, seeds will enter the indeterministic stage, in which AFL includes havoc and splice. In this stage, AFL randomly selects a sequences of mutation operators and assigns random location to mutate the seed.

There are many studies on mutation strategies for fuzzer. VUzzer [8] leverages data flow and control flow features to infer the critical regions of the input for mutation. GREYONE [60] uses a fuzzing-driven taint inference to infer taint variables for mutation. Superion [61] deploys mutation strategies to fuzz programs that process structured inputs. MOPT [14] uses particle swarm optimization algorithm to optimize mutation operators.

2.2.4. Power Schedule

Power schedule aims to allocate energy to each seed during the fuzzing process, which determines the number of seed mutations. Reasonable energy allocation can effectively improve the discovery of new paths. If the energy of a seed is over allocated, other seeds mutation will be affected. Conversely, if the energy of one seed is under allocated, it will be detrimental to new path discovery and potential bug detection.

AFL has two power schedule methods based on different mutation stages. In the deterministic stage, the energy of a seed is related to its length. The longer seed length, the more energy will be consumed. In the indeterministic stage, the energy allocation depends on the running time, the number of edges, the average size of the file, the number of cycles, and others.

Recent research shows that power schedule is very critical for fuzzer. AFLFast [11] allocates more energy to the low-frequency path to explore more paths. EcoFuzz [15] uses reinforcement learning to model power schedule as the adversarial multi-armed bandit model that enables adaptive energy saving. However, they did not consider the path risk and the effectiveness of energy allocation.

3. The Design of MooFuzz

To address problems mentioned in the previous sections, we propose a many-objective optimization fuzzer MooFuzz, as shown in Figure 2. The main components of MooFuzz contain static analyzer, feedback collector, seed scheduler, and power scheduler. In MooFuzz, static analyzer marks the risk edge and records the risk value for each edge by

scanning the source code and then inserts code fragments to update the edge risk value in running program. Feedback collector is used to record and update related information to guide the seed schedule after the program execution. Seed scheduler adopts different many-objective optimization schedules based on different states of the seed pool to select seeds. Power scheduler assigns energy based on feedback information and monitors energy usage.

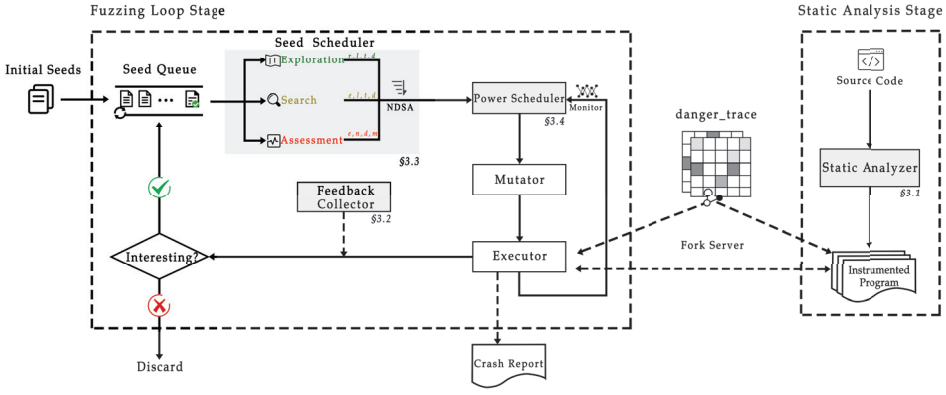


Figure 2. A high-level overview of MooFuzz.

3.1. Static Analyzer

A common idea is that the place has dangerous functions may trigger vulnerabilities. For example, the function *malloc* is used to dynamically allocate memory in C language. Although it can automatically allocate memory space, if used improperly, it may cause problems such as overflow, heap exhaustion, and use-after-free. The function *write* shall attempt to write n bytes from the buffer pointed to by buf into the file associated with the open file descriptor. However, if programmer cannot control the size of the bytes written to buf, it will cause the risk of out-of-bounds read of the memory. Therefore, MooFuzz identifies potentially dangerous functions as risk edges to label in static analyzer. In this paper, MooFuzz uses functions in Table 1 as dangerous functions [62], including memory allocation, memory recovery, memory operation, string operation, and file I/O operation. At the same time, users can also customize dangerous functions and add them to static analyzer for fuzzing.

Table 1. Dangerous functions.

Class	Description	Function Name
memory	memory allocation	*calloc*; *malloc*; *realloc*
	memory recovery	*free*
	memory operation	*memcpy*; *memmove*; *memchr*; *memset*; *memcmp*
string	string operation	*strcpy*; *strlen*; *strcat*; *strcmp*; *strspn*; *strcoll*; *strstr*; *strtok*
others	file I/O	*read*; *write*

Algorithm 2 shows the basic idea of MooFuzz instrumentation. Before the static analysis, there are well-known potentially dangerous functions. The static analyzer can identify them by traversing the source code and perform source code instrumentation at the corresponding edge position without running the program. MooFuzz uses a pointer *danger_trace* to record the hit-counts of the risk edge in shared memory after running

program every time. Specifically, MooFuzz first obtains each basic block information of the program, then identifies each call instruction and judges whether someone is dangerous (Lines 1–7). If any exists, the hit-counts will be updated and stored in the memory pointed to by *danger_trace* (Lines 8–11).

Algorithm 2: Code instrumentation

Input: the program P, a set of dangerous functions DF, a pointer variable *danger_trace*

Output: the instrumented program P'

```
1  MAP_SIZE = 2^16
2  for basic_block in P do
3      bool risk = false
4      for CallInstr in basic_block do
5          if CallInstr.CalledFuncName in DF then
6              risk = true
7              break
8      cur = Random(0, MAP_SIZE)
9      if risk=true then
10         danger_trace[cur ⊕ pre]++
11         pre = cur >> 1
```

3.2. Feedback Collector

The feedback collector is mainly used to continuously update seed information to assist seed schedule. For the running of the instrumented program, a series of running information would be updated for seeds. Algorithm 3 shows the process of information updating by feedback collector. It takes the seed queue Q and the pointer variable as inputs, and output is the seed queue Q' with new information. The new information includes the number of times the seed has been selected, the path frequency, the path risk, and the mutation information. Specifically, MooFuzz selects a seed s by using seed scheduler (see Section 3.3) and updates the number of times it has been selected (Lines 1–3). Then, it uses a mutation strategy to generate a new test case s' and executes the target program by using test case s' (Lines 4–5). Next, two pointer variables *danger_bits* and $edge_r$ are used to update the edge risk (Line 6). Here, *danger_bits* is obtained with the pointer variable *danger_trace*. The $edge_r$ records the risk of each edge. At the beginning, the edge corresponding to dangerous function has a maximum value, while those of the other edges are zero. Next, if the mutated test case produces new coverage, MooFuzz will calculate path risk value (Lines 7–8). Next, MooFuzz traverses each seed in the seed pool and determine whether its path is the same as the current path. If so, the frequency information of the seeds in seed pools will be updated (Lines 9–11). Finally, if the path of s is identical to the path of s', the mutation information will be updated (Lines 12–13).

We discuss how to update different information separately as follows.

The path risk mainly refers to the ability of seeds to detect dangerous locations, which determines the number and speed of bug discovery. Before discussing the path risk, we first give the definition of edge risk update and then that of path risk update.

The edge risk update. Given an edge e_i and the corresponding hit-count $danger\_bits[e_i]$, the edge risk $edge_r[e_i]$ is updated as follows.

$$edge_r[e_i] = \begin{cases} edge_r[e_i] - danger\_bits[e_i], & e_i \in danger\_edge \\ 0, & others \end{cases} \tag{5}$$

where *danger_edge* is the set of edges corresponding to dangerous function.

Algorithm 3: Information update

Input: a seed queue Q, a pointer variable $danger\_bits$, a pointer variable $edge_r$, the instrumented program P

Output: a seed queue Q' with new information

```
S = SeedSchedule(Q)
for s in S do
    s.select_num++
    s' ← Mutation(s)
    (trace_bits, danger_bits) ← Run_target(s')
    update_edge_risk(danger_bits, edge_r)
    if is_NewCoverage(P, s') then
        calculate_path_risk(edge_r)
    for s_i in seed pool do
        if the path of s_i is the same as that of s' then
            update_fre_info(s_i)
    if the path of s' is the same as that of s then
        update_mta_info(s)
```

The path risk update. Given a seed s and the risk values of all edges covered by the seed s, the path risk of seed s, $s.risk$ is calculated as follows.

$$s.risk = \sum_{i=1}^{N} \frac{edge_r[e_i]}{N} \tag{6}$$

The path frequency indicates the ability of the seed to discover a new path. As time goes by, there are high-frequency paths and low-frequency paths in the program. Generally, those seeds that cover low-frequency paths have a higher probability of discovering new paths than those that cover high-frequency paths (the larger the value, the higher the path frequency) after the program running for a while.

The path frequency update. Given a seed s' and its path $p_{s'}$, if there is a seed s in the seed pool and its path p_s, and p_s is the same as $p_{s'}$. We add one to the path frequency of seed s, that is,

$$s.fre = s.fre + 1, \text{if } p_{s'} = p_s \tag{7}$$

The mutation information indicates the mutation ability of a seed. For each seed that has not been fuzzed, its mutation effectiveness is set to 0, indicating that the seed has the best mutation validity. Among the seeds being fuzzed, the mutation ability of the seeds will be continuously evaluated, and individuals with high mutation ability (the smaller the value, the better) will obtain priority.

The mutation information update. Given a seed s and its mutation strategy M, if the path of seed s is the same as that of seed s' generated by seed mutation upon s, the mutation information of seed s, $s.mta$ is calculated as follows.

$$s.mta = s.mta + 1, \text{if } s' = M(s) \text{ and } p_{s'} = p_s \tag{8}$$

3.3. Seed Scheduler

Seed scheduler is mainly used for seeds selection. In order to effectively prioritize seeds, we propose a many-objective optimization seed schedule scheme.

Before seed schedule, MooFuzz divides the seed pool into three states according to seed attributes.

Exploration State. Exploration State refers to the existence of unfuzzed and favored seeds in the seed pool. Exploration State represents that the current seed pool state is an excellent state and it maintains the diversity of seeds.

Search State. In this state, the favored seeds have been fuzzed, but there are still unfuzzed seeds. Search State represents that there is a risk that the seed pool is completely fuzzed, and it is necessary to concentrate on finding more paths.

Assessment State. In this state, all the seeds are all fuzzed. It is very difficult to find a priority seed, but the fuzzed seeds produce a lot of information that can serve as a reference. Besides, MooFuzz performs state monitoring in the assessment state. Once the state changes, the seed set of the current state will be discarded to perform seed schedule in other states.

For these three states, MooFuzz uses different selection criteria based on bug detection, path discovery, and seed evaluation. MooFuzz constructs different objective functions based on different states.

In the previous discussion, MooFuzz has obtained the risk value of the seed before it is added to the seed pool, indicating the path risk. Based on previous research [8], seeds with deeper executing paths may be more capable of detecting bugs. Therefore, MooFuzz uses path risk r and path depth d as objectives for seed selection. To reduce the energy consumption of seeds and speed up the discovery of bugs, MooFuzz also takes the length l of the seed data and the execution time t of the seed as objectives. In Exploration State, MooFuzz uses the following objective functions to select the seeds that have not been fuzzed and favored.

$$Min\ F(s) = [-r, -d, l, t]^T, s \in S \tag{9}$$

Search State indicates that all the favored seeds in current seed pool have been fuzzed and there are unfuzzed seeds. At this time, MooFuzz's selection of seeds will mainly focus on the path discovery. The frequency information of the seeds will increase with the running time changes. In this state, those seeds that pass the low-frequency path will have greater potential to discover new paths. MooFuzz regards path frequency e and path depth d as criteria for seeds selection. Meanwhile, MooFuzz uses l and t described above to balance energy consumption. In Search State, MooFuzz uses the following objective functions to select the seeds that have not been fuzzed.

$$Min\ F(s) = [e, -d, l, t]^T, s \in S \tag{10}$$

Assessment State means that all seeds in the current seed pool have been fuzzed. MooFuzz will obtain the information of the seed including the path frequency e, the number of times that the seed has been selected n, the seed path depth d, and the mutation information m, and then add them to the objective functions as mutation criterion. Note that the current state does not choose the length and execution time of the seed as criteria to balance energy consumption, because the current state is very difficult to generate new seeds. Besides, once new seeds are generated in this state, Assessment State will be terminated and enter other state. In Assessment State, MooFuzz uses the following objective functions to select the seeds from the seed pool.

$$Min\ F(s) = [e, n, -d, m]^T, s \in S \tag{11}$$

MooFuzz selects the optimal seed set after establishing objective functions for different seed pool states and models seed schedule as a minimization problem. Algorithm 4 mainly completes the seed schedule by using non-dominated sorting [19]. The seed set S that satisfies state conditions will be selected as the input. A set CF that is used to store the optimal seed set. Initially, CF is an empty set, and s_1 in seed set S was added to CF. For each seed s_i from the seed set S and seeds s_j in CF finish the dominance comparisons (Lines 1–9). If s_j dominates s_i (each attribute value of s_j is less than s_i), the next seed comparison will be performed. If s_i dominates s_j, remove s_j from CF. After the comparison between the seed s_i and s_j, if there is not a dominance relationship between s_i and all the seeds in

CF, s_i will be added to CF (Lines 10–11). After the above cycle is completed, the optimal seed set is stored in CF, and MooFuzz extracts each seed inside for fuzzing (Lines 12–13).

Algorithm 4: Seed schedule

Input: the seed set S satisfying conditions in different states
Output: a series of optional seed s'

```
1  CF ← ∅
2  for s_i in S do
3      bool isdominated = false
4      for s_j in CF do
5          if s_j dominates s_i then
6              isdominated = true
7              break
8          if s_i dominates s_j then
9              CF ← CF − s_j
10     if not isdominated then
11         CF ← CF ∪ s_i
12 for s' in CF do
13     fuzz(s')
```

3.4. Power Scheduler

The purpose of power schedule is assigning reasonable energy for each seed involved in mutation. A high quality seed has more chances to mutation and should be assigned with more energy in fuzzing process.

Existing coverage-based fuzzers (such as AFL [7]) usually calculate the energy for the selected seeds as follows [18],

$$energy(i) = allocate\_energy(q_i) \tag{12}$$

where i is the seed and q_i is the quality of the seed, depending on the execution time, branch edge coverage, creation time, and so on.

Algorithm 5 is the seed power schedule algorithm. MooFuzz considers different seed pool states to set up different energy distribution methods. Meanwhile, it also uses an energy monitoring mechanism, which has the ability to monitor the execution of target programs and reduce unnecessary energy consumption.

After many experiments, we find that the amount of energy in the deterministic stage is mainly related to the length of the seed, which is a relatively fine-grained mutation, but as the number of candidate seeds in the seed pool increases, it will affect the path discovery. Thus, in Algorithm 5 we open the deterministic stage to seeds that cause crashes after mutation (Lines 1–2). In the indeterministic stage, MooFuzz judges the state of the current seed. If it belongs to Search State, MooFuzz uses the frequency information to set the energy. If it belongs to Assessment State, both the frequency and the mutation information will be comprehensively considered to set the energy (Lines 3–6).

After energy allocation, we set up a monitoring mechanism to monitor the mutation of seeds (Lines 7–14). When each seed consumes 75% of the allocated energy, MooFuzz monitors the mutation of the current seed, and records the ratio of the average energy consumption of the current seed covering a new path and that of all seeds covering a new path. If its ratio is lower than $threshold_1$, MooFuzz will withdraw the energy, if its ratio is higher than $threshold_2$, the mutation information will be updated. Here, $threshold_1$ is equal to 0.9 and $threshold_2$ is equal to 1.3.

Algorithm 5: Power schedule

Input: a seed s, the number of all seeds in seed pool *total_seed*, the total energy consumed in the fuzzing process *total_energy*, the number of new seeds generated by the current seed mutation *cur_seed*

Output: the energy of seed s $s.energy$

if seed s that causes crashes after mutation **then**
    goto deterministic stage
`/* indeterministic stage: */`
if state is Search State **then**
    $s.energy = (1 + \frac{1}{s.fre}) * energy(s)$
if state is Assessment State **then**
    $s.energy = (1 + (\frac{1}{s.mta} + \frac{1}{s.fre})) * energy(s)$
for $cur\_energy = 0$ to $s.energy$ **do**
    if the energy consumption of seed s reaches 75% **then**
        $total\_average = \frac{total\_energy}{total\_seed}$
        $cur\_average = \frac{cur\_energy}{cur\_seed}$
        if $\frac{cur\_average}{total\_average} < threshold_1$ **then**
            break
        if $\frac{cur\_average}{total\_average} > threshold_2$ **then**
            $s.mta = s.mta * 0.9$

4. Evaluation

MooFuzz is built on top of AFL-2.52b [7]. The implementation adds C/C++ code to the AFL. The instrumentation components are implemented to mark danger edges based on the LLVM framework [63] in static analysis. Through these experiments, the following research questions are tackled:

RQ1: How capable is MooFuzz in crash detection?
RQ2: How effective is the code coverage of MooFuzz?
RQ3: How capable is MooFuzz in identifying real-world vulnerabilities?

4.1. Experimental Settings

Baseline Fuzzer. We compare MooFuzz with existing state-of-the-art tools AFL [7], AFLFast [11], FairFuzz [13], and PerfFuzz [16]. The selection of baseline fuzzer is mainly based on the following considerations:

1. AFL is currently one of the most common coverage-based greybox fuzzer in community.
2. AFLFast is a variant of AFL with better power schedule.
3. FairFuzz is also an extending fuzzer of AFL. It optimizes inputs that hit rare branches.
4. PerfFuzz improves the instrumented components to generate pathological inputs.

Benchmark. To evaluate MooFuzz, we choose seven real-world open source Linux applications as the benchmark to conduct experiments. Jasper [64] is a software tool kit for processing image data that provides a way to represent images and facilitates the manipulation of image data. LibSass [65] is a C/C++ port of the Sass engine. Exiv2 [66] is a C++ library and a command line utility to read, write, delete, and modify Exif, IPTC, XMP, and ICC image metadata. Libming [67] is a library for generating Macromedia Flash files, written in C, and includes useful utilities for working with Flash files. OpenJPEG [68] is an open source JPEG 2000 codec written in C language. Bento4 [69] is a C++ class library that

is designed to read and write ISO-MP4 files. The GUN Binutils [70] is a collection of binary tools. Table 2 shows target applications and their fuzzing configure.

Table 2. Target applications and their fuzzing configure.

Program	Command Line	Project Version
jasper	jasper - -input @@ - -output t.bmp	Jasper 2.0.14
libsass	tester @@	LibSass 3.5.4
exiv2	exiv2 -pX @@	Exiv2 0.26
libming	listswf @@	Libming 0.4.8
openjpeg	opj_decompress -i @@ -o t.png	OpenJPEG 0.26
cxxfilt	c++filt -t	GUN Binutils 2.31
bento4	mp42hls @@	Bento4 1.5.1

Performance Metrics. Crashes, paths, and vulnerabilities are chosen as metrics in this section. In code coverage metrics, we use the number of seeds in the queue as an indicator and use tool Afl-cov [71] to measure code line coverage and function coverage. In vulnerability detection, we directly use AddressSanitizer [72] to detect it.

Experiment Environment. All experiments are conducted on a server configured with two Xeon E5-2680 v4 processors (56 logical cores in total) and 32 GB RAM. The server installed Ubuntu 18.04 system. For the same application, the initial seed set is the same. We fuzz each application for 24 h (on a single logical core) and repeat 5 times to reduce randomness. In all implementations, we use 42 logical cores, and we leave 14 logical cores for other processes to keep the workload stable.

4.2. Unique Crashes Evaluation (RQ1)

In order to evaluate the effectiveness of MooFuzz, a direct method is to evaluate the number of crashes and the speed at which they are triggered. It is believed that more crashes may trigger more bugs. We fuzz each application to run on 5 different fuzzers to compare the number of unique crashes and the speed of discovery. Figure 3 shows the growth trends of unique crashes discovery in different fuzzers. From these results, we can make the follow observations.

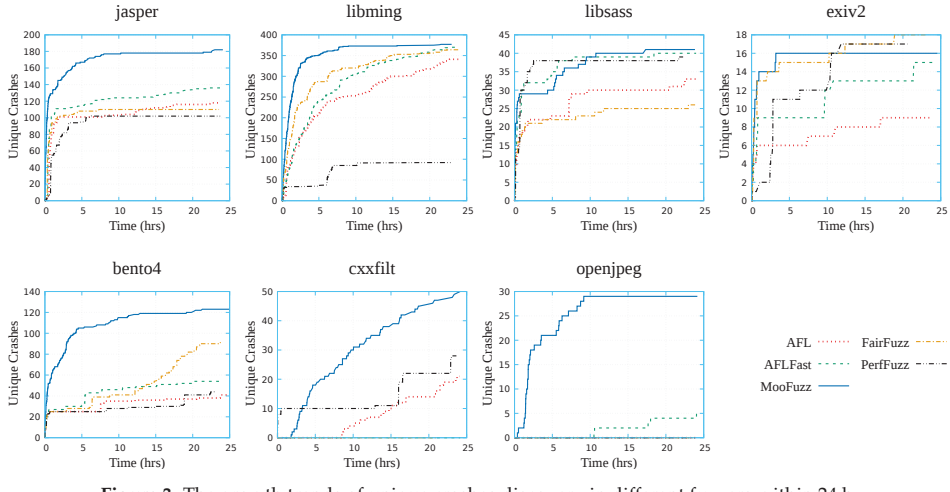


Figure 3. The growth trends of unique crashes discovery in different fuzzers within 24 h.

First, different fuzzers have different capability in fuzzing different application programs. For example, PerfFuzz has zero crash in fuzzing openjpeg within 24 h, but it can

trigger most crashes in fuzzing exiv2 among other fuzzers. This shows that the different criteria of the seed selection affect the number of crashes.

Second, seed schedule and power schedule affect the efficiency of crashes discovery. The experimental results show that MooFuzz outperforms AFL in the speed of crashes discovery and just takes about 10 h to trigger most of the unique crashes. There is no path risk measurement and energy monitoring in AFL, leading to a lot of time spent on invalid mutation operators.

Third, MooFuzz is able to find more crashes than other state-of-the-art fuzzers. The static results are shown in Table 3. We count the number of crashes found in applications by different fuzzers within 24 h, and count the total number of crashes found by each fuzzer. Table 3 shows that except for exiv2, MooFuzz triggers more crashes than other fuzzers, among which jasper triggers 182 crashes within 24 h and AFL only triggers 118 crashes. In total, MooFuzz triggers 818 crashes in benchmark application programs, improving by 46%, 32%, 34%, and 153%, respectively, compared with state-of-the-art fuzzers AFL [7], AFLFast [11], FairFuzz [13], and PerfFuzz [16].

Table 3. Number of unique crashes found in real-world programs by various fuzzers.

Program	MooFuzz	AFL	AFLFast	FairFuzz	PerfFuzz
jasper	182	118	136	110	102
libming	377	341	371	364	92
libsass	41	33	40	26	39
exiv2	16	9	15	18	17
bento4	123	40	54	91	44
cxxfilt	50	21	0	0	29
openjpeg	29	0	5	0	0
total	818	562	621	609	323

Overall, MooFuzz significantly outperforms other fuzzers in terms of speed and number of unique crashes.

4.3. Coverage Evaluation (RQ2)

Code coverage is an effective way to evaluate fuzzers. The experiment measures coverage from source code line, function, and path. Table 4 shows the line and function covered by different fuzzers. In total, MooFuzz's line coverage and function coverage are better than AFL, AFLFast, FairFuzz, and PerfFuzz.

Table 4. Line and function covered by fuzzers.

Program	MooFuzz		AFL		AFLFast		FairFuzz		PerfFuzz	
	Line	Func	Line	Func	Line	Func	Line	Func	Line	Func
jasper	32.8%	47.5%	32.1%	46.4%	32.2%	46.7%	31.4%	45.8%	32.2%	46.7%
libming	15.5%	16.8%	13.6%	14.8%	13.0%	14.3%	16.1%	17.3%	6.0%	7.4%
libsass	52.2%	35.0%	51.1%	35.1%	50.2%	34.5%	52.2%	35.3%	45.3%	32.8%
exiv2	5.0%	9.0%	4.9%	8.8%	4.9%	8.9%	4.9%	8.8%	4.9%	8.8%
bento4	12.1%	12.6%	11.4%	11.5%	11.4%	11.5%	11.5%	11.7%	11.5%	11.7%
cxxfilt	2.5%	3.0%	2.5%	3.0%	2.7%	3.1%	2.8%	3.1%	1.5%	2.5%
openjpeg	31.2%	41.4%	29.2%	33.5%	31.7%	41.3%	33.2%	41.9%	29.5%	39.0%

Figure 4 shows the growth trends of paths discovery in five different fuzzers after fuzzing applications for 24 h. We can clearly observe that except for cxxfilt, MooFuzz ranks first among all fuzzers from the perspective of the number of path discovery. Among them, it can find about 6000 paths in fuzzing openjpeg, and the other four fuzzers can only find about 3600 paths. It can find about 1200 paths after fuzzing jasper for 24 h, while other fuzzers can only find about 500 to 700 paths. Although the number of paths discovered

by MooFuzz is lower than FairFuzz and AFLFast in fuzzing cxxfilt, it can trigger the most crashes compared with other fuzzers. From the speed of path discovery, MooFuzz is significantly higher than other fuzzers.

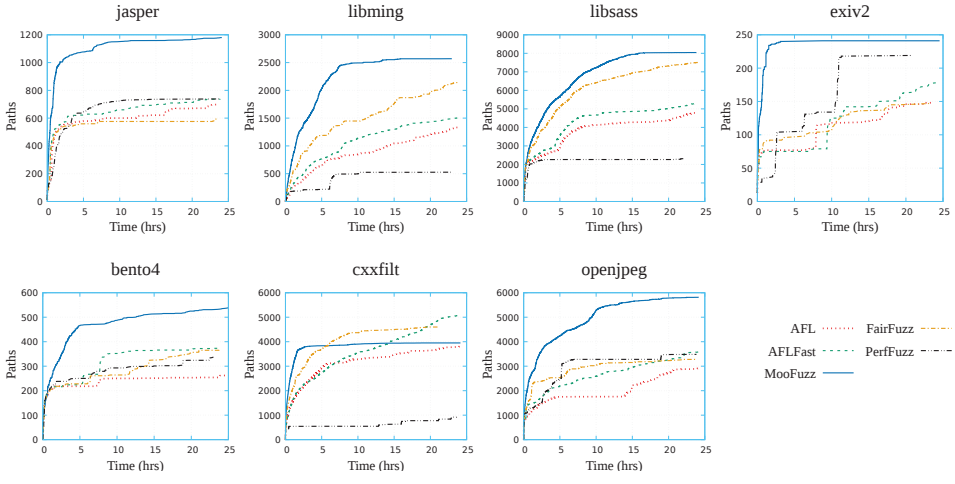


Figure 4. The growth trends of paths discovery in different fuzzers within 24 h.

Overall, MooFuzz outperforms other fuzzers in terms of line, function, and path coverage.

4.4. Vulnerability Evaluation (RQ3)

MooFuzz tests old version of the applications and analyzes related vulnerabilities to evaluate the ability in vulnerability detection. Table 5 shows the real vulnerabilities combination with its IDs identified by MooFuzz. MooFuzz is able to find stack overflow, heap overflow, null pointer dereference, and memory leaks related vulnerabilities.

Table 5. Real-world vulnerabilities found by MooFuzz.

Program	CVE	Vulnerability
cxxfilt	CVE-2018-9138	stack overflow
cxxfilt	CVE-2018-17985	stack overflow
jasper	CVE-2018-19543	out-of-bounds read
jasper	CVE-2018-19542	null pointer dereference
jasper	CVE-2018-19541	out-of-bounds read
libsass	CVE-2018-19837	stack overflow
libsass	CVE-2018-20821	stack overflow
libsass	CVE-2018-20822	stack overflow
exiv2	CVE-2018-16336	heap-buffer-overflow
exiv2	CVE-2018-17229	heap-buffer-overflow
exiv2	CVE-2018-17230	heap-buffer-overflow
exiv2	CVE-2017-14861	stack-buffer-overflow
libming	CVE-2018-13066	memory leaks
libming	CVE-2020-6628	heap-buffer-overflow
openjpeg	CVE-2020-8112	heap-buffer-overflow
bento4	CVE-2019-15050	out-of-bounds read
bento4	CVE-2019-15048	heap-buffer-overflow

Vulnerability analysis. We use a real-world application program vulnerability to analyze the effectiveness of our approach, as shown in Figure 5. This is a code snippet from openjpeg [68] which contains a heap-buffer-overflow vulnerability (i.e., CVE-2020-8112).

In Figure 5, the main function contains a conditional statement (Lines 1–9). In MooFuzz, the seed s_t satisfies the judgment condition and enters the true branch to execute function *opj_tcd_decode_tile*(...). Moreover, the seed s_n enters the false branch to execute other codes that do not contain dangerous functions. As*malloc* is a dangerous function which is used in *opj_tcd_decode_tile*(...), risks might emerge when this function is used. Therefore, MooFuzz preferentially selects seed s_t for mutation. In this case, *malloc*(*l_data_size*) is called and *l_data_size* comes from an unsigned operation in the function *opj_tcd_decode_tile*. Then, the function *opj_t*1*_clbl_decode_processor* will be called in the following program flow, where the allocated memory will be modify through two variables *cblk_h* and *cblk_w*. All of these two variables are obtained through signed operation, which causes an integer overflow making $cblk\_h * cblk\_w > l\_data\_size$, and MooFuzz easily satisfies the above conditions through mutation, so the heap-buffer-overflow happened.

```
1  int main(int argc, char **argv){
2  ...
3  if (!parameters.nb_tile_to_decode){
4  opj_tcd_decode_tile(...)
5  }
6  else{
7  ...
8  }
9  }
10
11 OPJ_BOOL opj_tcd_decode_tile(opj_tcd_t *p_tcd, ...){
12 /* unsigned operations */
13 OPJ_SIZE_T res_w = (OPJ_SIZE_T)(l_res->x1 - l_res->x0);
14 OPJ_SIZE_T res_h = (OPJ_SIZE_T)(l_res->y1 - l_res->y0);
15
16 l_data_size = res_w * res_h;
17 /* tile->data allocate l_data_size space */
18 tilec->data = malloc(l_data_size);
19 ...
20 opj_t1_clbl_decode_processor(...);
21 }
22
23 static void opj_t1_clbl_decode_processor(...){
24 /* cblk_h and cblk_w are obtained through signed operation
25 * which cause integer overflow
26 * cblk_h * cblk_w > l_data_size, heap overflow happened. */
27 datap = tilec->data;
28 for (j = 0; j < cblk_h; ++j){
29 i = 0;
30 for (; i < (cblk_w & ~(OPJ_UINT32)3U); i += 4U){
31 OPJ_INT32 tmp0 = datap[(j * cblk_w) + i + 0U];
32 ...
33 }
34 }
35 }
```

Figure 5. An example from openjpeg (CVE-2020-8112).

4.5. Discussion

We enhance fuzzing from the perspectives of vulnerabilities and coverage. Although more coverage may trigger more vulnerabilities, not all coverage is equal [62]. Based on our observation of the fuzzing process, we define the path risk and prioritize seeds that consume less energy while executing high risks, to maximize the improvement of fuzzing. Meanwhile, we use different objectives for seed optimization and energy allocation. It can improve the efficiency of fuzzing in a limited time.

In the algorithm design of the power schedule, we use two thresholds to judge the current seed energy usage. There is still an opportunity to adaptively adjust these two thresholds instead of the fixed thresholds. For example, these thresholds can be dynamically adjusted according to the fuzzing process. In our evaluation, our method can improve the probability of triggering vulnerabilities, but it may not be effective for triggering vulnerabilities that require complex conditions, such as deeply nested conditions. Although we use a variety of open source benchmarks to evaluation MooFuzz, it may not be effective for programs that require specific grammatical conditions for inputs (such as XML). However, the prototype we develop, MooFuzz, is a completely dynamic prototype.

It can integrate static analysis techniques like symbolic execution to generate test cases that satisfy specific conditions to improve fuzzing.

5. Conclusions and Further Work

In this paper, a many-objective optimization model is built for seed schedule. Considering the three states of the seed pool, we use different objective functions to select seeds from the perspectives of bug detection, path discovery, and seed evaluation. At the same time, an energy recovery mechanism is designed to monitor energy usage during the fuzzing process. We implement a prototype MooFuzz on top of AFL and evaluate it on seven real-world programs. The experiment results show that MooFuzz behaves more effectively than state-of-the-art fuzzers in path discovery and bug detection.

In the future, we plan to use MooFuzz to fuzz the latest version of the applications to assist testers in testing. In next study, we will consider optimizing power schedule through multi-information feedback on the basis of MooFuzz, so that it can monitor energy consumption according to the current program running progress, and automatically set and adjust energy. We also consider starting from the seed mutation, and propose a new decision model to determine effective region of the seed and select the effective mutation strategy.

Author Contributions: Conceptualization, X.Z. and H.Q.; methodology, X.Z. and H.Q.; software, W.L. and X.Z.; validation, W.L., S.L., and X.Z.; formal analysis, H.Q.; investigation, X.Z. and S.L.; resources, J.X.; data curation, W.L.; writing—original draft preparation, X.Z.; writing—review and editing, X.Z. and J.X.; visualization, W.L. and S.L.; supervision, H.Q.; project administration, X.Z.; funding acquisition, H.Q. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China grant number 61827810.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The test results data presented in this study are available on request. The data set can be found in public web sites.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Arya, A.; Neckar, C. Fuzzing for Security. Available online: https://blog.chromium.org/2012/04/fuzzing-for-security.html (accessed on 30 November 2020).
2. Evans, C.; Moore, M.; Ormandy, T. Fuzzing at Scale. Available online: https://security.googleblog.com/2011/08/fuzzing-at-scale.html (accessed on 30 November 2020).
3. Moroz, M.; Serebryany, K. Guided in-Process Fuzzing of Chrome Components. Available online: https://security.googleblog.com/2016/08/guided-in-process-fuzzing-of-chrome.html (accessed on 30 November 2020).
4. Godefroid, P.; Kiezun, A.; Levin, M.Y. Grammar-based whitebox fuzzing. In Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2008), Tucson, AZ, USA, 21–25 June 2008; pp. 206–215. [CrossRef]
5. Arkin, B. Adobe Reader and Acrobat Security Initiative. Available online: https://blogs.adobe.com/security/2009/05/adobe_reader_and_acrobat_secur.html (accessed on 30 November 2020).
6. Serebryany, K. Continuous fuzzing with libFuzzer and AddressSanitizer. In Proceedings of the 2016 IEEE Cybersecurity Development (SecDev 2016), Boston, MA, USA, 3–4 November 2016; p. 157. [CrossRef]
7. Zlewski, C. American Fuzzy Lop. Available online: http://lcamtuf.coredump.cx/afl (accessed on 1 September 2020).
8. Rawat, S.; Jain, V.; Kumar, A.; Cojocar, L.; Giuffrida, C.; Bos, H. VUzzer: Application-aware evolutionary fuzzing. In Proceedings of the 24th Annual Network and Distributed System Security Symposium (NDSS 2017), San Diego, CA, USA, 26 February–1 March 2017; pp. 1–14. [CrossRef]
9. Gan, S.; Zhang, C.; Qin, X.; Tu, X.; Li, K.; Pei, Z.; Chen, Z. Collafl: Path sensitive fuzzing. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (S&P 2018), San Francisco, CA, USA, 21–23 May 2018; pp. 679–696. [CrossRef]
10. Sun, L.; Li, X.; Qu, H.; Zhang, X. AFLTurbo: Speed up path discovery for greybox fuzzing. In Proceedings of the 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE 2020), Coimbra, Portugal, 12–15 October 2020; pp. 81–91. [CrossRef]

11. Böhme, M.; Pham, V.; Roychoudhury, A. Coverage-based greybox fuzzing as markov chain. *IEEE Trans. Softw. Eng.* **2019**, *45*, 489–506. [CrossRef]
12. Böhme, M.; Pham, V.T.; Nguyen, M.D.; Roychoudhury, A. Directed greybox fuzzing. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS 2017), Dallas, TX, USA, 30 October–3 November 2017; pp. 2329–2344. [CrossRef]
13. Lemieux, C.; Sen, K. Fairfuzz: A targeted mutation strategy for increasing greybox fuzz testing coverage. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE 2018), Montpellier, France, 3–7 September 2018; pp. 475–485. [CrossRef]
14. Lyu, C.; Ji, S.; Zhang, C.; Li, Y.; Lee, W.H.; Song, Y.; Beyah, R. MOPT: Optimized mutation scheduling for fuzzers. In Proceedings of the 28th USENIX Security Symposium (USENIX Security 2019), Santa Clara, CA, USA, 14–16 August 2019; pp. 1949–1966.
15. Yue, T.; Wang, P.; Tang, Y.; Wang, E.; Yu, B.; Lu, K.; Zhou, X. EcoFuzz: Adaptive energy-saving greybox fuzzing as a variant of the adversarial multi-armed bandit. In Proceedings of the 29th USENIX Security Symposium (USENIX Security 2020), Vancouver, BC, Canada, 12–14 August 2020; pp. 2307–2324.
16. Lemieux, C.; Padhye, R.; Sen, K.; Song, D. PerfFuzz: Automatically generating pathological inputs. In Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2018), Amsterdam, The Netherlands, 16–21 July 2018; pp. 254–265. [CrossRef]
17. Wen, C.; Wang, H.; Li, Y.; Qin, S.; Liu, Y.; Xu, Z.; Chen, H.; Xie, X.; Pu, G.; Liu, T. Memlock: Memory usage guided fuzzing. In Proceedings of the 42nd International Conference on Software Engineering (ICSE 2020), Han River, Seoul, Korea, 6–11 July 2020; pp. 765–777. [CrossRef]
18. Li, Y.; Xue, Y.; Chen, H.; Wu, X.; Zhang, C.; Xie, X.; Wang, H.; Liu, Y. Cerebro: Context-aware adaptive fuzzing for effective vulnerability detection. In Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE 2019), Tallinn, Estonia, 26–30 August 2019; pp. 533–544. [CrossRef]
19. Yuan, Y.; Xu, H.; Wang, B. An improved NSGA-III procedure for evolutionary many-objective optimization. In Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO 2014), Vancouver, BC, Canada, 12–16 July 2014; pp. 661–668. [CrossRef]
20. Rizk-Allah, R.M.; El-Sehiemy, R.A.; Deb, S.; Wang, G.G. A novel fruit fly framework for multi-objective shape design of tubular linear synchronous motor. *J. Supercomput.* **2017**, *73*, 1235–1256. [CrossRef]
21. Li, J.; Lei, H.; Alavi, A.H.; Wang, G.G. Elephant herding optimization: Variants, hybrids, and applications. *Mathematics* **2020**, *8*, 1415. [CrossRef]
22. Sun, J.; Miao, Z.; Gong, D.; Zeng, X.J.; Li, J.; Wang, G. Interval multiobjective optimization with memetic algorithms. *IEEE Trans. Cybern.* **2020**, *50*, 3444–3457. [CrossRef]
23. Wang, F.; Li, Y.; Liao, F.; Yan, H. An ensemble learning based prediction strategy for dynamic multi-objective optimization. *Appl. Soft Comput.* **2020**, *96*, 106592. [CrossRef]
24. Feng, Y.; Wang, G.G.; Li, W.; Li, N. Multi-strategy monarch butterfly optimization algorithm for discounted {0-1} knapsack problem. *Neural Comput. Appl.* **2018**, *30*, 3019–3036. [CrossRef]
25. Srikanth, K.; Panwar, L.K.; Panigrahi, B.K.; Herrera-Viedma, E.; Sangaiah, A.K.; Wang, G.G. Meta-heuristic framework: Quantum inspired binary grey wolf optimizer for unit commitment problem. *Comput. Electr. Eng.* **2018**, *70*, 243–260. [CrossRef]
26. Feng, Y.; Deb, S.; Wang, G.G.; Alavi, A.H. Monarch butterfly optimization: A comprehensive review. *Expert Syst. Appl.* **2021**, *168*, 114418. [CrossRef]
27. Pan, Q.K.; Sang, H.Y.; Duan, J.H.; Gao, L. An improved fruit fly optimization algorithm for continuous function optimization problems. *Knowl.-Based Syst.* **2014**, *62*, 69–83. [CrossRef]
28. Sang, H.Y.; Pan, Q.K.; Duan, P.y. Self-adaptive fruit fly optimizer for global optimization. *Nat. Comput.* **2019**, *18*, 785–813. [CrossRef]
29. Wang, F.; Li, Y.; Zhou, A.; Tang, K. An estimation of distribution algorithm for mixed-variable newsvendor problems. *IEEE Trans. Evol. Comput.* **2020**, *24*, 479–493. [CrossRef]
30. Wang, G.G.; Guo, L.; Gandomi, A.H.; Hao, G.S.; Wang, H. Chaotic krill herd algorithm. *Inf. Sci.* **2014**, *274*, 17–34. [CrossRef]
31. Wang, G.G.; Deb, S.; Cui, Z. Monarch butterfly optimization. *Neural Comput. Appl.* **2019**, *31*, 1995–2014. [CrossRef]
32. Gao, D.; Wang, G.G.; Pedrycz, W. Solving fuzzy job-shop scheduling problem using DE algorithm improved by a selection mechanism. *IEEE Trans. Fuzzy Syst.* **2020**, *28*, 3265–3275. [CrossRef]
33. Sang, H.Y.; Pan, Q.K.; Duan, P.Y.; Li, J.Q. An effective discrete invasive weed optimization algorithm for lot-streaming flowshop scheduling problems. *J. Intell. Manuf.* **2018**, *29*, 1337–1349. [CrossRef]
34. Wu, G.; Pedrycz, W.; Li, H.; Ma, M.; Liu, J. Coordinated planning of heterogeneous earth observation resources. *IEEE Trans. Syst. Man, Cybern. Syst.* **2015**, *46*, 109–125. [CrossRef]
35. Wang, G.G.; Gandomi, A.H.; Yang, X.S.; Alavi, A.H. A new hybrid method based on krill herd and cuckoo search for global optimisation tasks. *Int. J. Bio-Inspired Comput.* **2016**, *8*, 286–299. [CrossRef]
36. Wang, G.G.; Guo, L.; Duan, H.; Liu, L.; Wang, H.; Shao, M. Path planning for uninhabited combat aerial vehicle using hybrid meta-heuristic DE/BBO algorithm. *Adv. Sci. Eng. Med.* **2012**, *4*, 550–564. [CrossRef]
37. Yi, J.H.; Wang, J.; Wang, G.G. Improved probabilistic neural networks with self-adaptive strategies for transformer fault diagnosis problem. *Adv. Mech. Eng.* **2016**, *8*, 1–13. [CrossRef]

38. Mao, W.; He, J.; Li, Y.; Yan, Y. Bearing fault diagnosis with auto-encoder extreme learning machine: A comparative study. *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.* **2017**, *231*, 1560–1578. [CrossRef]
39. Mao, W.; Feng, W.; Liang, X. A novel deep output kernel learning method for bearing fault structural diagnosis. *Mech. Syst. Signal Process.* **2019**, *117*, 293–318. [CrossRef]
40. Wang, G.G.; Lu, M.; Dong, Y.Q.; Zhao, X.J. Self-adaptive extreme learning machine. *Neural Comput. Appl.* **2016**, *27*, 291–303. [CrossRef]
41. Mao, W.; Zheng, Y.; Mu, X.; Zhao, J. Uncertainty evaluation and model selection of extreme learning machine based on Riemannian metric. *Neural Comput. Appl.* **2014**, *24*, 1613–1625. [CrossRef]
42. Duan, H.; Zhao, W.; Wang, G.G.; Feng, X.H. Test-sheet composition using analytic hierarchy process and hybrid metaheuristic algorithm TS/BBO. *Math. Probl. Eng.* **2012**, *2012*, 1239–1257. [CrossRef]
43. Liu, G.; Zou, J. Level set evolution with sparsity constraint for object extraction. *IET Image Process.* **2018**, *12*, 1413–1422. [CrossRef]
44. Wu, G.; Pedrycz, W.; Suganthan, P.N.; Li, H. Using variable reduction strategy to accelerate evolutionary optimization. *Appl. Soft Comput.* **2017**, *61*, 283–293. [CrossRef]
45. Li, R.; Zheng, Q.; Li, X.; Yan, Z. Multi-objective optimization for rebalancing virtual machine placement. *Future Gener. Comput. Syst.* **2020**, *105*, 824–842. [CrossRef]
46. Srinivas, N.; Deb, K. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evol. Comput.* **1994**, *2*, 221–248. [CrossRef]
47. Coello, C.A.C.; Pulido, G.T.; Lechuga, M.S. Handling multiple objectives with particle swarm optimization. *IEEE Trans. Evol. Comput.* **2004**, *8*, 256–279. [CrossRef]
48. Felde, I.; Szénási, S. Estimation of temporospatial boundary conditions using a particle swarm optimisation technique. *Int. J. Microstruct. Mater. Prop.* **2016**, *11*, 288–300. [CrossRef]
49. Wang, F.; Zhang, H.; Zhou, A. A particle swarm optimization algorithm for mixed-variable optimization problems. *Swarm Evol. Comput.* **2021**, *60*, 100808. [CrossRef]
50. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
51. Yi, J.H.; Deb, S.; Dong, J.; Alavi, A.H.; Wang, G.G. An improved NSGA-III algorithm with adaptive mutation operator for Big Data optimization problems. *Future Gener. Comput. Syst.* **2018**, *88*, 571–585. [CrossRef]
52. Yi, J.H.; Xing, L.N.; Wang, G.G.; Dong, J.; Vasilakos, A.V.; Alavi, A.H.; Wang, L. Behavior of crossover operators in NSGA-III for large-scale optimization problems. *Inf. Sci.* **2020**, *509*, 470–487. [CrossRef]
53. Zhang, Q.; Li, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **2007**, *11*, 712–731. [CrossRef]
54. Wang, R.; Zhang, Q.; Zhang, T. Decomposition-based algorithms using pareto adaptive scalarizing methods. *IEEE Trans. Evol. Comput.* **2016**, *20*, 821–837. [CrossRef]
55. Wang, R.; Zhou, Z.; Ishibuchi, H.; Liao, T.; Zhang, T. Localized weighted sum method for many-objective optimization. *IEEE Trans. Evol. Comput.* **2018**, *22*, 3–18. [CrossRef]
56. Wang, G.G.; Tan, Y. Improving metaheuristic algorithms with information feedback models. *IEEE Trans. Cybern.* **2017**, *49*, 542–555. [CrossRef]
57. Ishibuchi, H.; Tsukamoto, N.; Nojima, Y. Evolutionary many-objective optimization: A short review. In Proceedings of the 2008 IEEE Congress on Evolutionary Computation (CEC 2008), Hong Kong, China, 1–6 June 2008; pp. 2419–2426. [CrossRef]
58. Petsios, T.; Zhao, J.; Keromytis, A.D.; Jana, S. Slowfuzz: Automated domain-independent detection of algorithmic complexity vulnerabilities. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS 2017), Dallas, TX, USA, 30 October–3 November 2017; pp. 2155–2168. [CrossRef]
59. Wang, H.; Xie, X.; Li, Y.; Wen, C.; Li, Y.; Liu, Y.; Qin, S.; Chen, H.; Sui, Y. Typestate-guided fuzzer for discovering use-after-free vulnerabilities. In Proceedings of the 42nd International Conference on Software Engineering (ICSE 2020), Han River, Seoul, Korea, 6–11 July 2020; pp. 999–1010. [CrossRef]
60. Gan, S.; Zhang, C.; Chen, P.; Zhao, B.; Qin, X.; Wu, D.; Chen, Z. GREYONE: Data flow sensitive fuzzing. In Proceedings of the 29th USENIX Security Symposium (USENIX Security 2020), Boston, MA, USA, 12–14 August 2020; pp. 2577–2594.
61. Wang, J.; Chen, B.; Wei, L.; Liu, Y. Superion: Grammar-aware greybox fuzzing. In Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE 2019), Montreal, QC, Canada, 25–31 May 2019; pp. 724–735. [CrossRef]
62. Wang, Y.; Jia, X.; Liu, Y.; Zeng, K.; Bao, T.; Wu, D.; Su, P. Not all coverage measurements are equal: Fuzzing by coverage accounting for input prioritization. In Proceedings of the 27th Annual Network and Distributed System Security Symposium (NDSS 2020), San Diego, CA, USA, 23–26 February 2020; pp. 1–17. [CrossRef]
63. Lattner, C.; Adve, V. LLVM: A compilation framework for lifelong program analysis & transformation. In Proceedings of the International Symposium on Code Generation and Optimization (CGO 2004), Palo Alto, CA, USA, 20–24 March 2004; pp. 75–86. [CrossRef]
64. Jasper. Available online: https://www.ece.uvic.ca/~frodo/jasper/ (accessed on 23 September 2020).
65. Libsass. Available online: https://sass-lang.com/libsass (accessed on 23 September 2020).
66. Exiv2. Available online: https://exiv2.org/ (accessed on 23 September 2020).
67. Ming. Available online: https://github.com/libming/libming (accessed on 23 September 2020).

68. Openjpeg. Available online: https://www.openjpeg.org/ (accessed on 23 September 2020).
69. Bento4. Available online: https://www.bento4.com/ (accessed on 23 September 2020).
70. Binutils. Available online: https://www.gnu.org/software/binutils/ (accessed on 23 September 2020).
71. Afl-cov. Available online: https://github.com/soh0ro0t/afl-cov (accessed on 23 September 2020).
72. Serebryany, K.; Bruening, D.; Potapenko, A.; Vyukov, D. AddressSanitizer: A fast address sanity checker. In Proceedings of the 2012 USENIX Annual Technical Conference (USENIX ATC 2012), Boston, MA, USA, 13–15 June 2012; pp. 309–318.

mathematics

MDPI

Article

Two-Population Coevolutionary Algorithm with Dynamic Learning Strategy for Many-Objective Optimization

Gui Li [1], Gai-Ge Wang [1,2,3,*] and Shan Wang [4,5]

[1] Department of Computer Science and Technology, Ocean University of China, Qingdao 266100, China; liguiatqingdao@gmail.com
[2] Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China
[3] Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Guangxi University for Nationalities, Nanning 530006, China
[4] Faculty of Arts and Humanities, University of Macau, Macau 999078, China; shanwang@um.edu.mo
[5] Institute of Collaborative Innovation, University of Macau, Macau 999078, China
* Correspondence: wgg@ouc.edu.cn

Abstract: Due to the complexity of many-objective optimization problems, the existing many-objective optimization algorithms cannot solve all the problems well, especially those with complex Pareto front. In order to solve the shortcomings of existing algorithms, this paper proposes a co-evolutionary algorithm based on dynamic learning strategy. Evolution is realized mainly through the use of Pareto criterion and non-Pareto criterion, respectively, for two populations, and information exchange between two populations is used to better explore the whole objective space. The dynamic learning strategy acts on the non-Pareto evolutionary to improve the convergence and diversity. Besides, a dynamic convergence factor is proposed, which can be changed according to the evolutionary state of the two populations. Through these effective heuristic strategies, the proposed algorithm can maintain the convergence and diversity of the final solution set. The proposed algorithm is compared with five state-of-the-art algorithms and two weight-sum based algorithms on a many-objective test suite, and the results are measured by inverted generational distance and hypervolume performance indicators. The experimental results show that, compared with the other five state-of-the-art algorithms, the proposed algorithm achieved the optimal performance in 47 of the 90 cases evaluated by the two indicators. When the proposed algorithm is compared with the weight-sum based algorithms, 83 out of 90 examples achieve the optimal performance.

Keywords: evolutionary algorithms (EAs); many-objective optimization; coevolution; dynamic learning; performance indicators

Citation: Li, G.; Wang, G.-G.; Wang, S. Two-Population Coevolutionary Algorithm with Dynamic Learning Strategy for Many-Objective Optimization. *Mathematics* **2021**, *9*, 420. https://doi.org/10.3390/math9040420

Academic Editor: Alicia Cordero

Received: 1 February 2021
Accepted: 19 February 2021
Published: 20 February 2021

1. Introduction

In recent years, with the development of technology, more and more new problems appear in the field of industry or control and so on. This problem is usually characterized by containing more than one objective function to be optimized, and these objective functions contradict each other. Such problems are generally called multi-objective optimization problems (MOPs) or many-objective optimization problems (MaOPs). Generally, MOPs are problems having two or three objectives, and MaOPs contain more than four objectives [1]. A MaOP is defined as follows:

$$\begin{aligned} &minimize \quad F(x) = \{f_1(x), f_2(x), \ldots, f_M(x)\} \\ &subject\ to \quad x \in X \end{aligned} \tag{1}$$

where M is the objective number. $x = (x_1, x_2, \ldots, x_n)$ is decision variable, and $X \subseteq R^n$ is the decision space of the n-dimensional real number field. F is a mapping from decision space to objective space, and F(x) contains M different objective functions $f_i(x)$ ($i = 1, 2, \ldots, M$).

Generally, the optimal solutions of MaOPs are distributed on Pareto front (PF), and the solutions on PF generally show the trade-off on all M objectives. Therefore, it is impossible to obtain an optimal solution by an optimization method to make it optimal on all objectives. What is needed to solve MaOPs is to obtain a set that has a finite number of solutions, so that the solutions in this solution set can well represent the whole PF, no matter in terms of convergence or diversity. The EAs had its unique advantages in solving MaOPs because of its population-based characteristics. After years of development, scholars from all over the world have proposed various many-objective optimization algorithms (MaOEAs) to solve MaOPs. Because different MaOPs also have different characteristics, currently no general algorithm can perfectly solve all MaOPs.

MaOPs usually have more than three objective numbers, so the objective space cannot be visualized, and the high-dimension calculation equation can only be obtained through derivation in solutions with fewer objectives. For example, grid-based evolutionary algorithm (GrEA) [2] deduces the high-dimensional grid calculation equation through a two-dimensional equation. Moreover, with the increasing of the objective number, the number of non-dominated individuals in the population will also increase exponentially. Studies have shown that almost all the solutions in the obtained population will be non-dominated when $M > 12$ [3]. As a result, the selection pressure of the algorithm based on non-dominated sorting is reduced, which makes the algorithm unable to solve the MaOPs well. In addition, the shape and density of PF vary greatly for different problems, which brings great challenges to obtaining a solution set with good convergence and diversity.

To overcome these difficulties with MaOPs, a number of MaOEAs have been proposed. For example, on the basis of fast and elitist multi-objective genetic algorithm (NAGA-II) [4], reference points are introduced to guide individual convergence and help evolution through the concept of domination, called NSGA-III [5,6]. Knee point-driven evolutionary algorithm (KnEA) [7] used knee points to guide individual convergence, and GrEA introduced the concept of grid to choose the better of two non-dominated individuals. In addition, some indicator-based algorithms are proposed, such as indicator-based selection in multi-objective search (IBEA) [8] and fast hypervolume-based algorithm (HypE) [9], which adopt $I_{\varepsilon+}$ [10] and hypervolume (HV) [11,12] indicators as the criteria for selecting individuals, respectively. The selection process is the evolution process of the whole population or individuals towards the direction with better indicator values. Finally, there is the evolutionary algorithm based on decomposition (MOEA/D) [13], which adopts the idea of mathematical decomposition to decompose a MaOP into M subproblems for simultaneous optimization. Common decomposition approaches include weighted sum approach, Tchebycheff approach, and penalty-based boundary intersection approach. There is a new way called MOEA/D-PaS [14] to combine the decomposition with Pareto adaptive scalarizing methods to balance the selection pressure toward the Pareto optimal and the algorithm robustness to PF.

At the same time, these MaOEAs also contain some disadvantages. The convergence speed of the Pareto-based algorithm is slow, or even unable to converge to PF. Indicator-based algorithms often tend to favor one or some of special regions of PF. Although the convergence speed of an indicator-based algorithm is generally relatively fast, the diversity of the solution set is usually poor. Besides, the decomposition-based MaOEAs are very dependent on the selection of decomposition approaches, such as weighted sum method in dealing with the non-convex problem (in the case of minimization) of PF shape, and not all Pareto optimal vectors can surely be obtained [13]. In addition, when the shape and density of PF are very complex and changeable, the traditional method to generate a weight vector is not suitable for this environment.

Existing multi-objective optimization algorithms (MOEAs) will still be affected by the increased number of objectives when dealing with MOPs, especially MAOPs. Moreover, the complexity of the PF also brings great challenges to the MOEAs. However, the dynamic learning strategy can pay more attention to the improvement of convergence when the population has poor convergence in the early evolutionary stage, and pay more attention

to the improvement of diversity in the late evolutionary stage. In addition, coevolution is a promising idea to improve the quality of individuals in a population through the mode of cooperation (or competition) between multiple populations (or subpopulations). Through coevolution, some key information of the population (such as the evolutionary state of the population) can be obtained in the process of algorithm iteration. The information of the population can be fed back flexibly to change the dynamic level (this will be discussed in Section 4) of dynamic learning strategies. In conclusion, the combination of a dynamic learning strategy and a coevolution model is a promising way to improve the convergence and diversity of the population.

The rest of this paper will be arranged as follows. Section 2 introduces the related works of the coevolutionary algorithm, other background, and the motivation for a two-population coevolution algorithm with dynamic learning strategy (DL-TPCEA). Section 3 will give the background of the MaOPs. The algorithm framework, process details, and parameter settings will be introduced in Section 4. Sections 5 and 6 carry out the analysis of experiments and the conclusion, respectively.

2. Related Works

In biology, the concept of coevolution is defined as follows: an adaptive coevolution in which two interacting species develop in the course of evolution. An evolutionary type of genetic evolution in which one species is influenced by another. At the biological level, it has several major significances:

1. Promote the increase of biological diversity;
2. Promote the co-adaptation of species;
3. Maintain the stability of the biological community.

This idea was successfully introduced into computer algorithms. More and more researchers begin to pay attention to the performance improvement brought by coevolution strategy to the EAs. In order to adapt to increasingly complex problems, Potter et al. incorporated the idea of coevolution into EAs. It extended the evolutionary paradigm of the time and described an architecture that evolved subcomponents into collections of collaborative species [15]. Then they analyzed the robustness of cooperative coevolutionary algorithms (CCEAs), which provided a theoretical basis for the effectiveness of coevolutionary strategies [16]. Wiegand et al. also used evolutionary game theoretic (EGT) models to help understand CCEAs and analyze whether CCEAs are really suitable for optimization tasks [17]. One of the EGT models was the multi-population symmetric game, which can be used to analyze and model the coevolutionary algorithm. In this context, coevolution tended to decompose an evolving population into several small subpopulations and ensured that each subpopulation did not interfere with each other. Then the individual in the population was optimized continuously through the cooperation of each subpopulation. The effectiveness of CCEAs is verified using CCEAs to solve complex problems (or structure) [18,19].

The coevolution strategy of CCEAs can group the population, which is suitable for large-scale optimization problems (LSOPs). The dimension of decision variables in LSOP is too high, so grouping is a good solution at present. This is also the initial application scenario of CCEAs. Yang et al. considered that traditional CCEAs can only deal with and decompose separable LSOPs, but often cannot solve the inseparable LSOPs. Therefore, a stochastic grouping scheme and adaptive weighting were introduced into problem decomposition and coevolution, and a new differential evolutionary algorithm was used to replace the traditional evolutionary algorithm. Through this improvement, the algorithm can effectively optimize the 1000-dimensional indivisible problem [20]. In addition, a multilevel coevolution (MLCC) [21] framework was proposed to solve LSOPs. MLCC was a framework that determined the size of a group when the problem was decomposed. MLCC constructed a set of problem decomposers based on random grouping strategies with different group sizes, and used an adaptive mechanism to select decomposers based on historical performance to self-adapt between different levels.

CCEAs is also applied to optimization problems in other scenarios. Liu et al. used cooperative coevolution (CC) to improve the speed of evolutionary programming (EP) [22]. However, this study showed that the time cost increased linearly as the dimension of the problems was increased. CC was also used to deal with global optimization and find global optimal solutions [23]. Chen et al. proposed a cooperative coevolution with variable interaction learning (CCVIL) framework [24], which treated all variables as independent and put them into separate groups, and then continuously merged groups when found the relationship between them at the iteration.

In addition to the above optimization problems, many researchers in recent years have begun to apply CC to MaOPs. Tan et al. combined SPEA2 and CC effectively and proposed SPEA2-CC [25]. After experimental comparison, the performance of SPEA2-CC was significantly better than that of the original SPEA2 as the number of objectives increases. SPEA2-CC provided theoretical support for the scalability of performance of CC in MaOPs.

A lot of researchers combined CC with the preference of the decision maker to deal with MaOPs, which led to the preference-inspired coevolutionary algorithm (PICEA) [26]. Researchers have shown that PICEA can handle not only MOPs, but also MaOPs [27]. The experiments showed that the preference-driven coevolution algorithm was superior to some other methods under the measurement of a hypervolume indicator. One defect of PICEA was the uneven distribution of the obtained solutions on PF, which means poor diversity. In order to solve this problem, an improved fitness allocation method (PICEA-g) [28] was proposed, which can consider the density information of solutions. In addition, a new preference-inspired coevolutionary algorithm using weight vectors (PICEA-w) [29] was proposed. The algorithm coevolved with the candidate solution during the search process. Coevolution adaptively constructed the appropriate weights in the optimization process, thus, effectively led the candidate solutions to the PF.

Liang et al. proposed a multi-objective coevolutionary algorithm based on a decomposition method [30], which used subpopulations to enhance objectives. Running on multiple subpopulations and external archive via the differential evolution (DE) operator to improve each objective and diversify the trade-offs of external archiving solutions. In addition, when an objective was not optimized, computing resources on that objective were allocated to other objectives and external archive strengthens the tradeoffs on all objectives. In addition, PF was approximated by parallel subpopulations [31]. Firstly, the MaOPs were decomposed by using a uniformly distributed weight vector, and then each subpopulation was associated with a weight vector. Using subpopulations to optimize each subproblem, and elite individuals in subpopulations were used to produce offspring. This can not only enhanced the diversity of the population, but also accelerated the convergence rate.

There were also studies that used new approaches to further improve CC performance on MaOPs. Shu et al. proposed a preference-inspired coevolution algorithm (PICEA-g/LPCA) with local principal component analysis (PCA) oriented goal vectors [32]. PICEA-g/LPCA was a further improvement on the basis of PICEA-g, and it used local PCA to extend the ability of PICEA-g and improved the convergence. In addition, a co-evolutionary particle swarm optimization algorithm with a bottleneck objective learning (BOL) strategy [33] was proposed to meet the convergence and diversity challenges in finite population size. In this algorithm, multiple subpopulations coevolved to maintain diversity. The BOL strategy was also used to improve convergence across all objectives. Elitist learning strategy (ELS) was also used to jump out of local PFs, and juncture learning strategy (JLS) was used to develop areas that are missing in PF.

Coevolution strategies have now been applied to many problems. In addition to general MaOPs, there are dynamic interval many-objective optimization problems (IMaOPs) [34,35], large-scale multi-objective optimization problems (LSMOPs) [33,36], and feature selection [37]. Finally, some recent work also uses coevolution or learning techniques [38,39] to deal with MaOPs [40–42].

As described in Section 1, the solution set obtained by Pareto-based MOEAs has a good distribution on PF, but there is a general problem of slow convergence and the performance will decline with the increase of the objective number. Non-Pareto MOEAs shows good convergence performance, but not good diversity performance. The solution set of non-Pareto MOEAs tends to converge to one or some special regions of PF, especially in the case of extremely irregular PF. Li et al. proposed a bi-criterion evolution (BCE) framework in 2015 [43], which performed well in many-objective optimization. In the BCE framework, two populations evolved simultaneously. One used the Pareto criterion (PC) and the other used the non-Pareto criterion (NPC). The aim was to take advantage of both approaches and compensate for their shortcomings. These two parts work together to promote evolution through the exchange of information between populations. Among them, NPC population led PC population to converge, and PC population can make up for the loss of NPC population in diversity. The two operations included in the framework, population maintenance and individual exploration, were used to preserve good non-dominated individuals and explore unexplored areas of NPC population respectively. Although the framework of BCE did not use the method of subpopulation coevolution in CC, the idea of cooperation between the two populations should also belong to CC.

Dynamic learning strategy (DLS) can consider the evolutionary state of solution set during algorithm iteration. It is well known that the initial population of MOEAs is randomly generated without specific requirements. The randomly generated solution is to take the value of the solution in the domain $[x_{\min}, x_{\max}]$ in the case of a normal (Gaussian) distribution. The equation is as follows:

$$x = x_{\min} + rand * (x_{\max} - x_{\min}) \tag{2}$$

where *rand* is a random number generated by a standard normal distribution. So, the convergence of initial population is very poor, just random individuals in the solution space. DLS can pay attention to this point, so that in the initial stage of population evolution, it can ensure rapid convergence of solutions by using more computing resources to the selection of convergence-related solutions. As the iteration goes on, the solutions converge towards PF. At this time, it is necessary to keep the solution set more diversified. Therefore, with the iteration of MaOEAs, computational resources will gradually incline to diversity-related solutions to maintain a better distribution of the population on PF.

Therefore, an effective combination of BCE and DLS may yield relatively good results, as confirmed by the experimental results in Section 5. This paper takes advantage of the coevolution of information interaction between the two populations, and introduces DLS into the environmental selection of NPC to better enable the evolution of NPC population. The cost value (CV) [44] will be selected as indicator. This algorithm will be called DL-TPCEA. The detailed algorithm will be described in Section 4.

3. The Background of MaOPs

At present, many single objective optimization problems in the optimization field have become the focus of research, such as workshop scheduling problems [45–49] and numerical optimization problems [50,51]. Most of these can be solved by classical algorithms and their improved versions, such as artificial bee colony algorithm (ABC) [47,48,52,53], particle swarm optimization (PSO) [51,54], monarch butterfly optimization (MBO) [55–58], ant colony optimization (ACO) [59,60], krill herd algorithm (KH) [52,61–64], elephant herding optimization (EHO) [65–67], and other metaheuristic algorithms [68–77]. However, there are some problems in many-objective optimizations, which cannot be solved by single objective techniques. Because of the conflicts between objectives, all objectives cannot be optimized simultaneously using single objective techniques. MaOPs also have different characteristics, which are described in more detail below. The current MaOPs in the field of many-objective optimizations are mainly divided into the following categories:

(1) General MaOPs: As mentioned in Equation (1), general MaOPs are problems with M conflicting objectives. The overall goal of solving MaOPs is to obtain a solution set that

can characterize PF, but at the same time there are a variety of problems. For objective numbers, MaOPs are more difficult to resolve than MOPs. Low dimensional optimization is mainly solved by non-dominated sorting, such as NSGA-II [4] and improving the strength of the Pareto evolutionary algorithm (SPEA2) [78]. The non-dominated sorting is described as follows: for the minimization problem, taking two vectors x_1 and x_2 in Ω, if and only if $f_i(x_1) \leq f_i(x_2)$ for each i in $\{1, 2, \ldots, M\}$ and $f_j(x_1) < f_j(x_2)$ for at least one j in $\{1, 2, \ldots, M\}$. Let us call it $F(x_1)$ Pareto dominates $F(x_2)$, and the notation is $F(x_1) > F(x_2)$, and if, and only if, no point x in Ω to satisfy $F(x) > F(x^*)$, called $F(x^*)$ Pareto optimal solution and x^* is Pareto optimal point, and the set of all Pareto optimal solutions is PF mentioned above, the set of all Pareto optimal points is called Pareto Set (PS).

(2) Large-scale MaOPs: these problems often involve high dimensional decision variables. In general, MOPs are called large-scale MOPs (LSMOPs) [79] when its decision variable dimension $N > 100$. The performance of the MaOEAs will decrease as the number of decision variables increases. For example, when using a mutation operator to mutate individuals, the probability of producing good individuals after mutation will also decrease due to the large dimension of decision variables. There are some researches on LSMOPs. At present, most of this work is based on classifying decision variables and dealing with them separately.

Ma et al. proposed a many-objective evolutionary algorithm based on decision variable analysis (MOEA/DVA) [80], which divided the whole population into convergence-related variables and diversity-related variables through decision variable analysis strategy. Moreover, MOEA/DVA optimized the two parts respectively, so that the convergence and diversity of the population were maintained well. Zhang et al. [81] proposed an evolutionary algorithm based on decision variable clustering for large-scale many-objective optimization problems (LMEA). LMEA used *k*-means clustering method and takes the angle between solutions and the direction of convergence as the feature to carry out the clustering, and divided the decision variables into convergence-related variables and diversity-related variables. LMEA further classified the unclassified individuals in MOEA/DVA to promote the convergence and diversity of the population. In addition, Chen et al. [1] proposed an evolutionary algorithm based on covariance matrix adaptation evolution strategy and scalable small subpopulation to solve large-scale many-objective optimization problems (S^3-CMA-ES).

The above work is based on the premise of grouping decision variables to deal with large-scale many-objective optimization problems, which makes a great contribution to the large-scale many-objective optimization.

(3) Dynamic MaOPs (DMaOPs): DMaOPs add time (environment) variation to the general MaOPs. It is described as follows:

$$\begin{aligned} \textit{minimize} \quad & F(x) = \{f_1(x, \mathrm{t}), f_2(x, t), \ldots, f_M(x, t)\} \\ \textit{subject to} \quad & x \in X \end{aligned} \tag{3}$$

where t is time (environment) variation. When time (environment) changes, PF of the MaOPs also changes, that is, the optimal solution set in the previous state is not necessarily the optimal solution set in the current state. This means that the algorithm is not only required to adapt to the many-objective environment to optimize multiple objectives, but also needs the changes brought by the response time (environment). When the time (environment) changes, the algorithm can respond quickly and get the optimal solution set in the latest environment.

In the environment of DMaOPs, many excellent algorithms have been proposed. Liu et al. proposed a dynamic multi-population particle swarm optimization algorithm (DP-DMPPSO) based on decomposition and prediction [82]. Using the archive update mechanism based on the objective space decomposition and the population prediction mechanism to accelerate the convergence, the results show that the algorithm has a good effect in DMaOPs processing. Finally, there are also many dynamic multi-objective evo-

lutionary algorithms (DMOEAs) that use various optimization strategies [83–87] to deal with DMaOPs.

The main purpose of this paper is to solve the general MaOPs with high dimensional objective space, using the Pareto-based and non-Pareto-based methods for coevolution of the two populations, respectively. The two populations make use of the advantages of each other and make up for the disadvantages, which is very promising to solve the difficulty of optimization in the high-dimensional objective space. The details will be introduced in Section 4.

4. The Framework of DL-TPCEA

In this part, the specific process of the dynamic learning strategy will be introduced first, and then the DL-TPCEA will be introduced. All algorithmic details such as parameter control and algorithmic flow are given.

4.1. Dynamic Learning Strategy

4.1.1. The Description of DLS

Previous MOEAs generally used the immutable evolution strategy during iteration. For example, NSGA-II used non-dominated sorting to select the non-dominated solutions in population to control the convergence, and then used crowding distance to select among the non-dominated solutions to improve the diversity of the population. This method is very time-consuming because of Pareto sorting, and tends to have poor convergence effect when the number of objectives is relatively large. However, DLS will make full use of the advantages of fast running speed and good convergence effect of indicator-based algorithm. Moreover, the enhancement of diversity is further strengthened to balance the convergence and diversity of the solution set. This paper will take a two-objective problem as an example to illustrate the advantages of DLS over traditional immutable evolutionary strategies.

As shown in Figure 1a, after the population initialization, the distribution of these individuals in the objective space is very chaotic. In other words, the convergence and diversity of the population are poor. According to the current population, the priority is to get these individuals to converge to PF as soon as possible. This will be guided by indicator-based method. For example, in a practical engineering problem, the individuals on PF are those who can minimize the cost. In this case, more computing resources should be allocated to the process of convergence-related operations to achieve rapid convergence of the population to PF. A small part of the computational resources are then allocated to operations that increase the diversity of the population to ensure that the diversity of the population is not particularly poor.

After the above operation, the distribution of individuals in the population in the objective space will gradually move towards PF, as shown in Figure 1b. However, the convergence level of the whole population is not enough at this time, so high selection pressure is still needed to promote convergence. As the iteration goes on, the distribution of individuals in population in the objective space will gradually become close to PF, as shown in Figure 1c. As introduced in Section 1, the indicator-based algorithm converges quickly but loses diversity easily. The example in Figure 1c shows that these individuals are close to PF under the guidance of the indicator, but the convergence position is more inclined to the central region of PF. At this point, more computing resources need to be tilted to increase the diversity of the population, such as the preference to keep the individual A and the individual B in Figure 1c into the next generation. By changing the computational resource allocation according to the evolutionary state of the population, individuals in the population can maintain good convergence and diversity. As in Figure 1d, the individuals in the resulting solution set are uniformly distributed on PF.

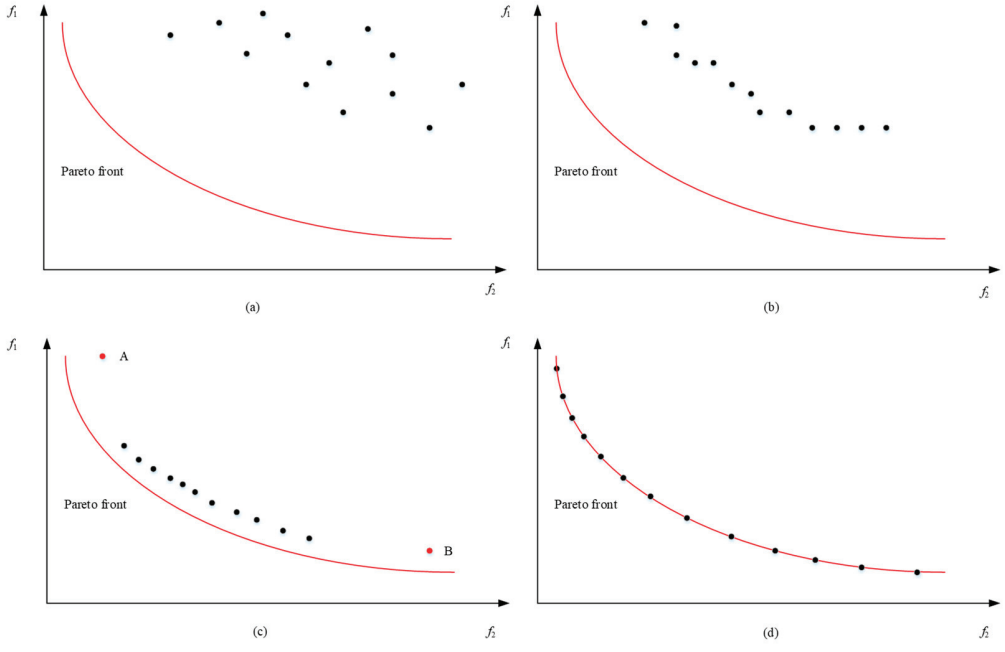


Figure 1. The process of dynamic learning strategy. (**a**) The state after initialization; (**b**) the state at the beginning of evolution; (**c**) the state of late evolution; (**d**) the state at the end of evolution.

4.1.2. The Details of DLS

The above is only a brief description of the steps of DLS; the following is a detailed explanation of the specific process of DLS. First, supposing the population size of MaOEAs is N. In an iteration, N new individuals are generated by crossover and mutation operators, at which time the original individuals and newly generated individuals form a new population, which is denoted as P_{2N} here. What needs to be done next is to select N individuals that are most conducive to maintain convergence and diversity through environmental selection as the initial population P_{new} of the next iteration. These operations are accomplished through DLS.

As shown in Algorithm 1, the $2N$ individuals are first layered by non-dominated sorting (Line 1, Algorithm 1). Here, *FrontNo* is the number of layers that each individual resides in, and *MaxFNo* is the largest number of layers that are non-dominated. Where *MaxFNo* satisfies:

$$\sum_{i=1}^{MaxFNo-1} L_i \leq N \&\& \sum_{i=1}^{MaxFNo} L_i > N \tag{4}$$

where L_i represents the number of individuals in the ith non-dominated layer ($i = 1, 2, \ldots,$ *MaxFNo*). Here, the non-dominated individuals in Layer 1 to layer *MaxFNo*-1 will preferentially select into P_{new} (Line 2, Algorithm 1), and then continue to select the remaining individuals in Layer *MaxFNo*.

Although in the case of 2- or 3-objective problems (MOPs), it may be more clearly layered. This makes the number of individuals in Layer *MaxFNo* smaller, which means fewer individuals are selected through DLS. However, with the increase of the objective number, the proportion of non-dominated individuals in the whole population also increased, almost all individuals are non-dominated when the objective number is more than 12 which is described in Section 1. This leads to an increase in the number of individuals in

Layer *MaxFNo*, even if all individuals in the population are in Layer *MaxFNo*. This also makes the role of DLS greatly increased, and become more useful in solving MaOPs.

Algorithm 1 Dynamic Learning Strategy

Input: Population: P_{2N}; Convergence factor: α; Parameter of p-norm: p
Output: Next generation population : P_{new}
1: $[FrontNo, MaxFNo] = NDSort()$
2: $P_{new} = P_{2N}(FrontNo < MaxFNo)$
3: Calculate C_n by Eq. (5)
4: Calculate D_n by Eq. (6)
5: Calculate the CV indicator of individuals by Eqs. (7) and (8)
6: Calculate the $L_p - norm$ distance between individuals
7: **for** $i = 1 : C_n$ **do**
8: Choose the individual by CV and put into P_{new}
9: **end for**
10: **for** $i = 1 : D_n$ **do**
11: Choose the individual by $PNormDis$ and put into P_{new}
12: **end for**

Next, the values of C_n and D_n will be calculated according to needs (Lines 3–4, Algorithm 1), representing the number of convergence-related individuals and diversity-related individuals that need to be preserved, respectively. This is also the key for DLS to ensure dynamic computational resource allocation within algorithm iteration. The calculation equation of the C_n is as follows:

$$C_n = \left\lceil R_{gen} \times \alpha \times \left(1 - \frac{gen}{max\ gen}\right) \right\rceil \tag{5}$$

where *gen* represents the current number of iterations and *maxgen* represents the maximum number of iterations. R_{gen} represents the total number of individuals that need to be selected at Layer *MaxFNo* at generation *gen*. Moreover, $\alpha \in [0, 1]$ is a convergence factor that controls the rate of convergence of the population. Through experimental research, it is found that when α is about 0.9, the performance can reach the best. In this way, the convergence speed can be achieved quickly at the same time; it will not fall into the local optimal. The symbol $\lceil \cdot \rceil$ rounds the element to the nearest integer greater than or equal to that element. Then the number of diversity-related individuals that needs to be preserved will continue to be calculated. The calculation of D_n is as follows:

$$D_n = R_{gen} - C_n \tag{6}$$

After C_n and D_n are calculated, two indicators of convergence and diversity will be calculated for the individuals in population, and R_{gen} individuals in Layer *MaxFNo* will be retained according to the rules of DLS. In this paper, cost value (*CV*) [44] will be selected as the convergence-related indicator. Let $F(x_i) = (f_1(x_i), f_2(x_i), \ldots, f_M(x_i))$ be the objective vector for individual x_i ($i = 1, 2, \ldots, N$). Then the mutual evaluation of individual x_i by individual x_j is as follows:

$$cv_{ij} = \max_m \{f_m(x_j)/f_m(x_i)\},\ m = 1, 2, \ldots, M \tag{7}$$

Then the mutual evaluation of each individual in the population is as follows:

$$CV_i = \min_{j \neq i} cv_{ij}, j = 1, 2, \ldots, N \tag{8}$$

This indicator will not be affected by the change of objective number, and the characteristics of this indicator can be clearly understood according to Equations (7) and (8). The first point is that x_i is a non-dominated individual when $CV_i > 1$, and the second is that x_i is a dominated individual when $CV_i \leq 1$. Therefore, we can use this indicator as convergence-related indicator to select the individuals in Layer *MaxFNo*. The individuals that have larger *CV* will be retained, in other words, retaining the individuals who have better convergence.

As for the diversity-related indicators, the distance between individuals in the population is generally used as the evaluation criterion. For example, Euclidean distance is used to calculate the crowding distance in NSGA-II. In this paper, the L_p-norm-based distance is selected to calculate the distance between individuals in the population. It has been experimentally demonstrated that the L_p-norm-based distance is more efficient than the Euclidean distance, Manhattan distance, etc., especially when dealing with MaOPs [88]. Parameter p of L_p-norm-based distance is recommended as $1/M$. Therefore, L_p-norm-based distance is selected as the diversity-related indicator in this paper.

After the calculation of two indicators for individuals in the population, the individuals in Layer *MaxFNo* were selected and saved to P_{new} according to C_n and D_n, until the size of P_{new} reached N.

4.2. The Framework of BCE

There are two main populations in BCE, namely NPC population and PC population. These two parts use the non-Pareto method and Pareto method to evolve the population, respectively. For the NPC population, any non-Pareto evolutionary criterion can be used directly. However, when the next generation is produced through competition, the environmental selection needs to select individuals from both NPC population and PC population (NPC selection). For PC population, non-dominated individuals from NPC and PC population are reserved (PC selection). Since the number of non-dominated solutions is unknown, population maintenance operation is carried out to eliminate some individuals with poor diversity when the number of non-dominated individuals is greater than the predefined threshold N.

Because of NPC population convergence speed is relatively fast, the individuals in NPC population can accelerate the convergence of PC population in PC selection. Because of the diversity of PC population is better, NPC population can explore the unexplored areas on PF through individual exploration operation and use the individuals in NPC population to enhance the diversity of PC population. In this way, the two populations interact with each other to promote the evolution of each population so that the convergence and diversity of the final solution set are good. The final output here is the PC population.

4.2.1. PC Selection and NPC Selection

The process of PC selection is to select non-dominated individuals from the mixed set of PC population and the new individuals produced by PC and NPC evolutions. NPC selection is based on the criteria of NPC evolution, which conducts environmental selection on a mixture of NPC populations and new individuals generated by PC populations. Assuming that the evolution of NPC populations uses indicator-based algorithms, then NPC selection is the selection of individuals having better indicator values in the mixed population for the next generation.

For the evolution of NPC populations, some algorithms rely on the information of the parent generation to update the individual, which is not feasible here. So individuals in the PC population are compared with individuals in the NPC population. If an individual in the PC population is better than one or more individuals in the NPC population according to the evolutionary criteria of the NPC population, then that individual (or a random one of those individuals) in the NPC population will be replaced by that individual in the PC population.

4.2.2. Population Maintenance

In PC selection, all non-dominated individuals are selected from a hybrid population of new individuals resulting from PC and NPC evolutions. Therefore, it is likely to make the number of non-dominated individuals larger than the preset threshold N (population size), especially when the objective number is large. Therefore, an effective means of population maintenance should be added to ensure that the PC population maintains a representative (with better convergence and diversity) group of individuals.

Population maintenance is to ensure the quality of individuals in a population through niche techniques. This is also a popular technique in EAs to assess the crowding degree of each individual in the population by the location and number of individuals in the niche (objective space). The crowding degree of individual p in population P is defined as follows:

$$D(p) = 1 - \prod_{q \in P, q \neq p} R(p, q) \tag{9}$$

$$R(p, q) = \begin{cases} d(p, q)/r, & if \quad d(p, q) \leq r \\ 1, & otherwise \end{cases} \tag{10}$$

where $d(p, q)$ is the Euclidian distance between individuals p and q, and r is the radius of the niche. Due to the size of each objective is different, in order to prevent the influence of problem size, the objective value of individuals in population will be normalized by maximum and minimum normalized first when calculating the distance.

It means that each is in the other's niche when the Euclidean distance between individuals p and q is less than r. This point can be seen in Equations (9) and (10), and the range of this crowding degree $D(p)$ is [0, 1]. Otherwise, there would be no effect on the crowding of these two individuals since $R(p, q) = 1$. When $d(p, q) \leq r$, the larger the Euclidean distance between the two individuals is, the smaller the calculated crowding degree will be, which means that the two individuals have a good crowding degree. So, this is a good way to eliminate the more crowded (poor diversity) individuals in the population.

Since the population is constantly evolving, it is not appropriate to set a fixed niche radius r. The setting of r must be related to the evolutionary state of the population. The radius r of the niche in BCE was set as the average Euclidian distance from each individual to k closest individuals in the population. The aim is to include one or more individuals in the niche of as many individuals as possible. Here, k is recommended to be set to 3 for better performance. Based on this crowding degree, the most congested individual in the population (the population of non-dominated individuals selected by PC selection) is eliminated each time and the crowding degree is recalculated. This process is repeated until the number of remaining non-dominated individuals is N.

4.2.3. Individual Exploration

The evolution part of the NPC population in BCE usually has high selection pressure; it converges quickly. However, the general NPC evolution tends to converge to one or more regions of PF, rather than the entire PF. This leads to a lack of diversity, as there are areas of PF that have never even been explored. It is through individual exploration that NPC evolution explores unknown areas on PF to achieve the purpose of increasing the diversity of NPC population. Individual exploration will explore some promising individuals in the PC population rather than all individuals in the PC population, because some individuals in the PC population have been well explored by NPC populations. These promising individuals generally have been eliminated (by NPC evolution), are less developed, or are not even visited in NPC evolution. From this point of view, the discussion is mainly focused on two types of individuals in the PC population:

1. Individuals whose niche has no NPC individual;
2. Individuals whose niche has only one NPC individual.

First of all, for the first group of individuals, these individuals are not in the niche of individuals in the NPC population. Such individuals are far away from the individuals in

the NPC population in the objective space, obviously not the individuals favored by the NPC criterion. However, it is such individuals that are in areas that NPC evolution has not explored. While the second kind of individuals have an NPC individual in niche, which is not a lot of individuals when considering that the k is set to 3. However, such individuals are likely to be located in areas where NPC evolution is incomplete, and it is necessary to explore such promising individuals.

During individual exploration, the above two kind of individuals contained in the PC population are first marked and stored in set S (individual sets to be explored). Then, the variation operation is carried out on the individuals in set S, and all the new individuals generated by the variation operator are stored in set T (the new individuals set generated by individual exploration) for the next PC selection. The variation operator here can be selected arbitrarily, but it should be noted that the number of parent individuals required by the selected variation operator should be changed accordingly.

The influence of the radius of the niche should also be considered here. A relatively small radius may allow all individuals in the PC population to be explored, as there may not be many NPC individuals in each individual's niche. The reverse is also true, larger radius may cause all individuals to remain unexplored. Therefore, a dynamically varying radius is used here, which can vary with the size of the PC population.

With the continuous evolutions of PC and NPC, more and more non-dominated individuals are produced, and the selection pressure of PC gradually decreases. This slows down PC evolution when the number of newly created non-dominated individuals exceeds the size of the remaining PC population that can be stored. This allows for less individual exploration, allowing the high selection pressure of NPC to play a greater role. The dynamic radius of the niche is set as follows:

$$r = (N\prime/N) * r_0 \tag{11}$$

where N represents the PC population size, and N' represents the size of the PC population before population maintenance, and r_0 represents the base niche radius calculated by means of population maintenance.

In the case of fixed computational resources (functional evaluations), this process of adaptive exploration is necessary according to the evolutionary state of the population. On the one hand, individual exploration can make up for the lack of diversity in NPC population. On the other hand, when there is a lack of convergence, more computing resources can be given to NPC evolution to accelerate convergence under higher selection pressure.

As shown in Figure 2, individual exploration on a 3-objective optimization problem is given. The triangle of coordinates in the figure represents the Pareto front of the problem, and the points in the figure represent the distribution of individuals in the population in the objective space. Suppose the NPC population is shown in Figure 2a, and the PC population is shown in Figure 2b. Due to the characteristics of NPC population, the obtained solution set may be distributed in some part of the Pareto front. For example, the population in Figure 2a is concentrated to the left and to the top of the Pareto front, while there is no individual distribution on the right. While PC population is relatively evenly distributed around the Pareto front, but the convergence is not good (some points do not converge to the Pareto front). Moreover, the role of individual exploration is to explore the promising individuals in the PC population to promote the diversity of the NPC population. It can be seen here that several individuals marked in red in Figure 2b are still promising individuals although they have not converged to the Pareto front. By exploring these solutions, it is possible to get some solutions that have never been explored in the PC population but have a good diversity. After continuous individual exploration, the diversity of PC population will also be improved and finally reach the state, as shown in Figure 2c. The population in Figure 2c well balances convergence and diversity, thus, achieving the purpose of individual exploration.

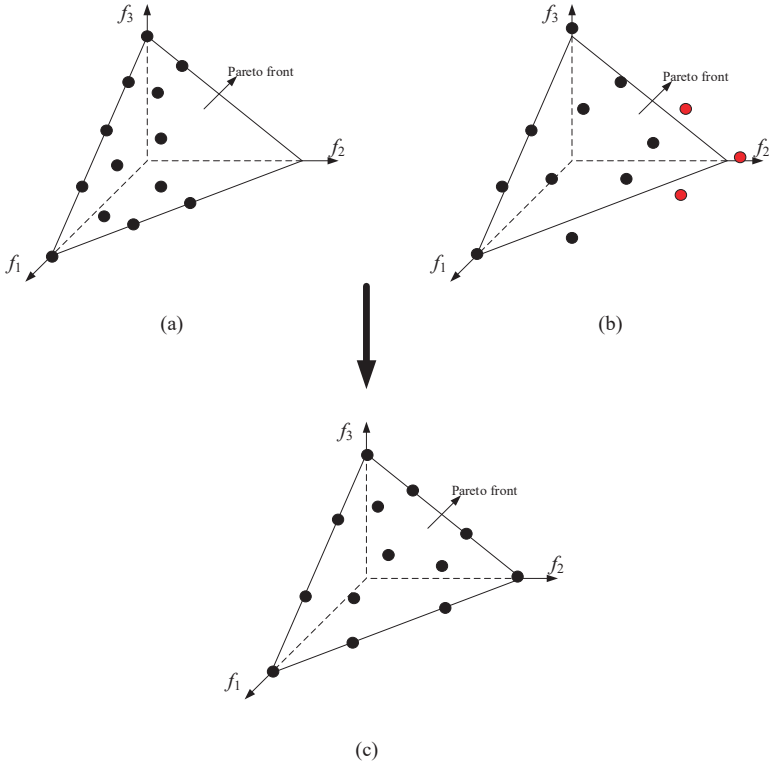


Figure 2. The individual exploration process in bi-criterion evolution (BCE). (**a**) The optimal solution set obtained by non-Pareto criterion; (**b**) the optimal solution set obtained by Pareto criterion; (**c**) the optimal solution set obtained by the individual exploration.

4.3. Two-Population Coevolutionary Algorithm with Dynamic Learning Strategy

From the above description, it can be seen that these strategies have great advantages and far-reaching significance in solving many-objective optimization problems. Next, we will introduce DL-TPCEA in the above context.

4.3.1. The Process of DL-TPCEA

Algorithm 2 gives the whole process of DL-TPCEA, from which it can be seen that the input parameters of DL-TPCEA include population size N, objective number M, and function evaluations (*FEs*). The final output is the population in which PC evolution. First, a parameter setting (Line 1, Algorithm 2) will be performed, which is mainly set for the current iteration number *gen* and the maximum iteration number *maxgen* in Equation (5). In addition, L_p-norm-based distance is also used in DLS for diversity maintenance, where the value p is also initialized. The inverse of the objective number ($1/M$) will be used here as the value of p. The parameter settings (Line 5, Algorithm 2) in the later steps do the same thing.

Before the proceeding of BCE, the populations (PC population and NPC population) in both evolutionary approaches should first be initialized (Lines 2–3, Algorithm 2). The NPC population randomly generates N decision vectors with dimension D in the domain by satisfying the normal distribution, where D represents the dimension of the decision variable. The PC population is generated by PC selection on the NPC population. This ensures that the individuals stored in the PC population will always be non-dominated.

Algorithm 2 Framework of DL-TPCEA

Input: Population size: N; The number of objective: M; Function evaluations: FEs
Output: The final population : PC
1: $ParameterSet()$
2: $NPC = Initialization()$
3: $PC = PCSelection(NPC)$
4: **while** $NotTermination(PC)$ **do**
5: $ParameterSet()$
6: $[NewPC, ExRatio] = Exploration(PC, NPC)$
7: $NPC = EnvironmentalSelection([NPC, NewPC], ExRatio, p)$
8: $NewNPC = Variation(NPC)$
9: $NPC = EnvironmentalSelection([NPC, NewNPC], ExRatio, p)$
10: $PC = PCSelection([PC, NPC, NewNPC])$
11: **end while**

When the algorithm begins to iterate, the individual exploration (Line 6, Algorithm 2) described in Section 4.2.3 is first performed. Exploring whether there are individuals in the PC population that the NPC population has not been (fully) explored. If these individuals existed, it will be stored in set *S* as described above. Then, the new individuals generated using variation operator to *S* was store in the set *T*. Finally, the returned NewPC population is all individuals in the set *T*. The *ExRatio* is a ratio coefficient, which represents the proportion of individuals to be explored. The *ExRatio* is calculated as follows:

$$ExRatio = \frac{Length(S)}{Length(PC)} \tag{12}$$

where *Length*(·) represents the size of the set or population. When *ExRatio* is greater than 0, it indicates that there are individuals in the PC population that need to be explored. The larger *ExRatio* means the more individuals in PC population need to be explored, and the value range of *ExRatio* is in [0, 1).

The *ExRatio* is set to dynamically change the convergence factor (dynamic convergence factor) of DLS later when using DLS for environment selection. As new individuals are generated by individual exploration, most of these individuals are located in areas that have not been explored or are not fully explored in NPC evolution. Therefore, the exploration at this iteration should pay more attention to these individuals, which means more diversity-related individuals should be appropriately selected to better explore these regions in the evolution of NPC. In this case, the convergence factor is appropriately scaled down according to the size of *ExRatio* at this iteration to achieve this purpose. The detailed process is described in Section 4.3.2.

After individual exploration, the following is the evolution of NPC population (Lines 7–9, Algorithm 2) and PC population (Line 10, Algorithm 2), respectively. First of all, an environment selection is carried out, and the individuals in mixed population of NewPC population and NPC population is selected by using the non-Pareto criterion and stored in NPC population. The variation operator is then applied to the NPC population to generate a new NewNPC population. Then the individuals with better performance in non-Pareto criterion are selected from the mixed population of NPC population and NewNPC population. The evolution of PC population uses PC selection to select non-dominated individuals in mixed population of original PC population, NPC population, and NewNPC population. This will select all non-dominated individuals from the three populations to archive in the PC population. Population maintenance operation is performed on the PC population if necessary (when *Length*(PC) > *N*).

4.3.2. Environmental Selection in NPC Evolution

The process of environmental selection in NPC evolution is shown in Algorithm 3. The environmental selection mainly uses DLS to select NPC population. However, dynamic convergence factors α' should be set according to the evolutionary state of the current population before selection. As the number of individuals explored by individual exploration is different at each iteration, the value of *ExRatio* is also different. However, when individuals need to be explored, the convergence factor α should be scaled down. In order to respond to the information of the number of individuals to be explored, the dynamic convergence factor α' is calculated as follows:

$$\alpha\prime = \alpha - \omega * \sin(\frac{ExRatio * \pi}{2}) \tag{13}$$

where ω is a dynamic scaling factor and is set to 0.1. The main purpose of this setting is to prevent the convergence factor from scaling too much, because a good convergence performance can be maintained when the convergence factor is set at 0.9 or so. Since the value interval of *ExRatio* is [0, 1), the value interval of dynamic convergence factor α' is [0.8, 0.9). This allows DLS to play a better role even in individual exploration.

After the predefined parameters are set, the next step is to select the individuals in the candidate population using DLS as shown in Algorithm 1. The population, here, is generated by the BCE process rather than a hybrid population with a parent–child relationship. In addition, the convergence factors used are dynamic convergence factors α' that are scaled according to the state of individual exploration.

Algorithm 3 Environmental Selection in NPC Evolution

Input: The population to be selected: P; Dynamic convergence factor ratio: $ExRatio$; Parameter of p-norm: p
Output: The final population : $NewNPC$

```
//The convergence factor α is set dynamically according to ExRatio
α = 0.9; ω = 0.1
if ExRatio > 0 then
    Calculte α' by Eq. (13)
else
    α' = α
end if
//The population was selected using DLS
NewNPC = DLS(P, α', p)
```

4.3.3. The Time Complexity Analysis of DL-TPCEA

The time complexity of DL-TPCEA is mainly determined by the party that consumes more time during the evolution of PC and NPC. In PC selection, the time complexity of selecting non- dominated individuals from the three-part population (Line 10 of Algorithm 2) is $O(MN^2)$. The time complexity of population maintenance and individual exploration is also $O(MN^2)$. So, the time complexity of PC evolution is $O(MN^2)$. In the NPC evolution, the time complexity of first non-dominated sort is $O(N\log^{M-2}N)$. The time complexity of calculating the number of convergence-related and diversity-related individuals is C (C is a constant), while the time complexity of calculating the two indicators of candidate solutions is $O(MN^2)$ and $O(N^2)$, respectively. The time complexity of using the indicators to select the candidate solution is $O(N)$. In conclusion, the time complexity of DL-TPCEA is $\max\{O(N\log^{M-2}N), O(MN^2)\}$.

5. Experiments

This section will verify the performance of the DL-TPCEA through experiments. First of all, the proposed dynamic convergence factor will be through a number of experiments to get an optimal equation. In addition, this paper will conduct an experimental analysis of the role of individual exploration in the whole evolutionary process. Finally, the performance of the DL-TPCEA is validated against several state-of-the-art algorithms.

5.1. Parameter Setting

In order to give full play to the performance of MaOEAs on MaOPs with different objective number, different *FEs* and population size *N* should be set for different objective number *M*. Taking the WFG [89,90] test suite as an example, the number of dimensions *D* needs to be dynamically changed. Here is set as recommended $D = M + 9$. In addition, the number of objectives in the experiments conducted in this paper is divided into five groups, and the number of objectives is 3, 5, 8, 10, and 15, respectively. In terms of population size setting, since reference points are used in both MOEA/D-PaS and NSGA-III, the original reference points need to be generated in a certain way. In this case, Das and Dennis's approach [91] is used to generate the original reference points on the hyperplane, while the other algorithms should have the same initial population size to ensure fairness. In addition, the number of generated reference points is the same with set in NSGA-III [5,6]. So, the corresponding population size *N* is set to 91, 210, 156, 275, and 135, respectively. The corresponding number of *FEs* is 10^4–$10^4 \times 5$. The detailed parameter settings are shown in Table 1.

Table 1. The parameter settings of experiments.

M	*N*	*D*	*FEs*
3	91	12	10000
5	210	14	20000
8	156	17	30000
10	275	19	40000
15	135	24	50000

In the experiments of dynamic convergence factor, α in the base DLS are set to the recommended 0.9. In the setting of dynamic convergence factors, various functions monotonically increasing in the interval [0, 1] are adopted for dynamic adjustment, which will be described in detail in Section 5.2. For all comparative algorithms in experiments, the parameter settings on each objective were also consistent with those in Table 1.

In addition, the running device is PC, the system version is Windows 10 enterprise version, the processor is Intel(R) Core (TM) i3-8100 CPU 3.6 GHz, and the RAM is 8 GB.

5.2. Experiments on Dynamic Convergence Factors

This paper proposes the concept of dynamic convergence factor in Section 4.3.2. The main approach is to determine the size of convergence factor dynamically based on the basic DLS and the state of individual exploration. The purpose of dynamic convergence factor is to make DLS adapt better to the evolutionary state of the population, so as to achieve the optimal convergence factor setting. Since the optimal value range of the convergence factor α is the interval [0.8, 0.9], we take the value of a monotone increasing function in the interval [0, 1] as shown in Equation (13), multiply it by a dynamic scaling factor ω by the function mapping of proportional coefficient *ExRatio*, and then subtract the corrected value from the original. In this way, it is possible to dynamically change the value of α' according to the proportionality coefficient *ExRatio*. In addition, when the *ExRatio* is relatively large (more diversity-related solutions need to be explored), the convergence factor can be appropriately reduced to better satisfy the convergence and diversity balance of the population.

In order to give full play to the optimal performance of the DL-TPCEA, certain work require to select the monotone increasing function in the interval [0, 1] of Equation (13). Columns 3 through 11 of the first row in Table 2 show some common monotone increasing functions in the interval [0, 1] as a comparative experiment. For example, the corresponding formula of Tan in the fourth column is as follows:

$$\alpha\prime = \alpha - \omega * \tan(\frac{ExRatio * \pi}{4}) \tag{14}$$

Table 2. The results of various monotone increasing functions on the WFG test suite, and the inverted generational distance (IGD) values of the results are tested by Friedman test.

M/Func	Ori	Sin	Tan	x^1	$x^{1/2}$	$x^{1/3}$	$x^{1/M}$	x^2	x^3	x^M	Fix
3	5.56	4.44	6.22	6.22	5.67	6.78	6.78	5.22	7.33	6.67	5.11
5	5.44	5.56	6.44	6.89	7.33	4.22	5.67	6.67	7.22	4.11	6.44
8	7.22	5.22	6.44	5.11	4.72	4.89	5.78	6.50	6.44	8.22	5.44
10	4.78	4.89	5.33	5.89	6.56	6.17	6.28	7.44	6.00	6.67	6.00
15	6.00	5.00	6.33	6.00	6.44	6.33	5.89	5.67	6.00	6.11	6.22
Avg	5.80	5.02	6.16	6.02	6.14	5.68	6.08	6.30	6.60	6.36	5.84

In addition, column 2 corresponds to the original DLS that the value of α is set to 0.9. The last column is a control group, and the method used here is that when $ExRatio > 0$, the value of α is multiplied by a value less than 1 (set as 0.9 here). This is equivalent to setting up a fixed set of transformations instead of making dynamic changes through functional mapping and the proportional coefficient *ExRatio*. This group is designed to analyze and compare the advantages and disadvantages of fixed transformations over functional mappings.

According to the parameter settings in Table 1, the different algorithms were run independently 30 times on each WFG test suite. The average inverted generational distance (IGD) values of the 30 runs were performed using the Friedman test (the smaller the better) and presented in Table 2. The last row is the average of the five sets of Friedman tests. Dark gray represents the best result and light gray represents the second-best result. From the results, the best performance is obtained when the monotone increasing function is taken as the sine function, and the optimal value (minimum value) is obtained on the 3- and 15-objective WFG, respectively. The second-best result is obtained on the 10-objective WFG. Although the results on 5- and 8-objective WFG are not so good, they are also relatively small in terms of numerical values. At the same time, the sinusoidal results were also the best among the average Friedman results of the five experiments. In addition, when the monotone increasing function is $x^{1/3}$, it performs second-best, and obtains second best results on the 5- and 8-objective WFG, respectively, and also obtains second best results in the average Friedman test.

When the monotone increasing function is selected as x^M and $x^{1/2}$, the optimal value is obtained on 5- and 8-objective WFG, respectively. However, the average Friedman test results for these two functions are not very good. It is worth noting that the original DLS obtained the optimal result on 10-objective WFG, followed by the mapping of sine function. In addition, the set of fixed transformations also showed the second-best result on 3-objective WFG. The average Friedman test results of these two strategies are not much different, but they are not as good as the average Friedman test results of the sine function.

Therefore, the sine function mapping of *ExRatio* is finally selected in dynamic convergence factor. In the experiments in Section 5.3 below, the dynamic convergence factor in DL-TPCEA is calculated in the form shown in Equation (13).

5.3. Experiments on Comparative Algorithms

To verify the performance of the proposed DL-TPCEA, we compare it with five state-of-the-art algorithms: MOEA/D-PaS [14], NSGA-III [5], CMOPSO [92], Two_Arch2 [88], and DLEA. The brief introduction to these comparative algorithms is given below.

In MOEA/D-PaS, a Pareto adaptive scalarizing (PaS) approximation method was proposed, which approximated the optimal p value of the commonly used scalarizing method. This is the key to balancing Pareto optimal selection pressure and algorithm robustness to PF geometries. It guarantees that any solution can be found along PF for given some weight. PaS is combined with the decomposition-based algorithm (MOEA/D) to increase the ability of balanced convergence and diversity.

NSGA-III is an improved version based on the framework of NSGA-II [4]. The crowding distance operator that was used to balance diversity in NSGA-II is modified into a diversity keeping strategy based on weight vector guidance. NSGA-III used a set of pre-generated uniformly distributed weight vectors to simulate the distribution of PF. When selecting solutions, the candidate solutions with the shortest vertical distance to these weight vectors will be selected.

CMOPSO is an improved version of the multi-objective particle swarm optimization (MOPSO [93]) by adding a competition mechanism. CMOPSO makes particles pairwise competitions to select particles in each generation of population. This makes the performance of CMOPSO less dependent on global and local optimal particles stored in an external archive.

Two_Arch2 uses two external archives, where each archive promotes convergence (CA) and diversity (DA). The two archives use different selection principles, where CA is indicator-based and DA is Pareto-based. At the same time, L_p-norm-based diversity maintenance scheme was also proposed in Two_Arch2 to improve the diversity of the population.

DLEA mainly uses the DLS mentioned in Section 4.1. The algorithm mainly used DLS to enhance the balance of convergence and diversity in environmental selection. Two different indicators are used to improve the performance by maintaining the convergence and diversity, respectively. Meanwhile, the convergence factor α in DLEA was fixed at 0.9. Compared with DLEA, DL-TPCEA proposes the concept of dynamic convergence factor. The comparison of these two algorithms is mainly to highlight the performance improvement brought by the dynamic convergence factors and the coevolution of the two populations.

The five comparative algorithms selected have their own characteristics, including operators frequently used in the many-objective optimization field: decomposition-based operator, Pareto-based operator, indicator-based operator, external-archive-based operator, and weight-vector-based operator. Comparing these algorithms can show the performance advantage of an algorithm more significantly.

For DL-TPCEA and other five comparative algorithms, Tables 3 and 4 give the mean and standard deviation (in parentheses) of HV values run on five sets of WFG test suites, and the results in Tables 5 and 6 are corresponding IGD values. Wilkerson Rank-Sum test ($\alpha = 0.05$) was used to test the significant difference between HV values and IGD values of these six algorithms. The symbols $-$, $+$, and $\approx$ stand for that the indicator values of the comparative algorithms were significantly worse than, better than, and similar to that of DL-TPCEA, respectively. In addition, for each test instance, the best (maximum) HV value and the best (minimum) IGD value were highlighted in gray.

Table 3. HV results of six algorithms on benchmarks WFG1-WFG9 with 3, 5, and 8 objectives.

Problems	M	MOEA/D-PaS	NSGA-III	CMOPSO	Two_Arch2	DLEA	DL-TPCEA
WFG1	3	1.7315e+1 (6.80e−1) −	2.0767e+1 (1.14e+0) −	1.1761e+1 (1.64e+0) −	1.9660e+1 (9.47e−1) −	2.4915e+1 (1.85e+0) −	2.7839e+1 (1.96e+0)
	5	7.8150e+2 (4.47e+2) −	2.8379e+3 (2.59e+2) ≈	1.6437e+3 (4.05e+1) −	2.2898e+3 (1.67e+2) −	2.8062e+3 (3.01e+2) ≈	2.6535e+3 (2.70e+2)
	8	5.4564e+6 (3.92e+5) −	6.5768e+6 (5.24e+5) ≈	5.2121e+6 (1.31e+5) −	6.4175e+6 (4.79e+5) ≈	8.2665e+6 (6.57e+5) +	6.9852e+6 (1.19e+6)
WFG2	3	5.7446e+1 (3.91e−1) −	5.7559e+1 (3.20e−1) −	5.8959e+1 (1.26e−1) ≈	5.7029e+1 (6.97e−1) −	5.8160e+1 (2.90e−1) −	5.8855e+1 (2.83e−1)
	5	5.2927e+3 (3.37e+2) −	6.0271e+3 (3.39e+1) −	6.0413e+3 (3.69e+1) −	5.9674e+3 (4.52e+1) −	5.9873e+3 (1.95e+1) −	6.1122e+3 (1.83e+1)
	8	1.0172e+7 (8.73e+6) ≈	2.1759e+7 (2.38e+5) ≈	2.1181e+7 (1.19e+5) −	2.1522e+7 (1.66e+5) −	2.1645e+7 (6.42e+4) −	2.1889e+7 (8.02e+4)
WFG3	3	6.0818e+0 (1.00e−1) +	5.5451e+0 (1.47e−1) −	5.7894e+0 (8.46e−2) −	5.5665e+0 (1.38e−1) −	6.1565e+0 (5.93e−2) +	5.9102e+0 (9.30e−2)
	5	1.1540e+0 (6.16e−2) ≈	1.2770e+0 (3.47e−1) ≈	4.1522e−1 (2.29e−1) −	1.2222e+0 (2.80e−1) ≈	1.8798e+0 (1.53e−1) +	1.1142e+0 (3.37e−1)
	8	1.4804e−2 (1.05e−4) +	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
WFG4	3	3.1857e+1 (3.32e−1) −	3.3687e+1 (2.16e−1) −	3.2170e+1 (1.79e−1) −	3.3801e+1 (2.25e−1) −	3.4028e+1 (2.89e−1) −	3.5106e+1 (1.47e−1)
	5	2.7725e+3 (2.50e+2) −	4.6140e+3 (2.49e+1) −	4.1442e+3 (4.18e+1) −	4.3112e+3 (3.61e+1) −	4.2991e+3 (5.25e+1) −	4.8905e+3 (2.30e+1)
	8	1.7243e+7 (2.76e+5) −	1.8176e+7 (2.42e+5) −	1.6073e+7 (5.95e+5) −	1.5929e+7 (5.49e+4) −	1.7267e+7 (1.02e+5) −	2.0259e+7 (1.20e+5)
WFG5	3	3.2283e+1 (1.20e−1) −	3.2008e+1 (1.51e−1) −	3.2193e+1 (3.26e−1) −	3.1499e+1 (2.29e−1) −	3.2310e+1 (1.84e−1) −	3.3072e+1 (2.45e−1)
	5	2.5150e+3 (2.93e+2) −	4.4522e+3 (2.15e+1) −	4.0531e+3 (1.17e+2) −	4.0890e+3 (3.42e+1) −	4.0815e+3 (5.67e+1) −	4.5798e+3 (1.63e+1)
	8	1.6273e+7 (2.15e+5) −	1.7208e+7 (1.66e+5) −	1.4956e+7 (9.07e+5) −	1.4988e+7 (1.39e+5) −	1.6070e+7 (3.91e+5) −	1.8993e+7 (7.80e+4)
WFG6	3	2.8635e+1 (2.28e+0) −	3.0041e+1 (7.45e−1) −	3.3213e+1 (3.90e−1) +	2.9430e+1 (7.41e−1) −	3.0771e+1 (7.68e−1) −	3.1280e+1 (8.81e−1)
	5	3.4025e+3 (3.89e+2) −	4.2268e+3 (1.19e+2) −	3.9875e+3 (1.69e+2) −	3.8312e+3 (9.55e+1) −	3.8672e+3 (1.54e+2) −	4.3856e+3 (8.73e+1)
	8	1.6134e+7 (5.09e+5) −	1.6452e+7 (7.46e+5) −	1.3440e+7 (5.23e+5) −	1.3529e+7 (2.04e+5) −	1.5082e+7 (3.53e+5) −	1.8534e+7 (4.34e+5)
WFG7	3	3.3709e+1 (3.69e−1) −	3.4017e+1 (2.20e−1) −	3.4256e+1 (1.77e−1) −	3.4185e+1 (2.61e−1) −	3.4551e+1 (2.13e−1) −	3.5763e+1 (9.24e−2)
	5	2.4688e+3 (3.46e+2) −	4.6163e+3 (3.97e+1) −	4.0731e+3 (8.89e+1) −	4.4358e+3 (4.30e+1) −	4.3903e+3 (4.88e+1) −	4.9534e+3 (8.29e+0)
	8	1.4711e+7 (3.57e+6) −	1.7534e+7 (3.70e+5) −	1.5266e+7 (4.98e+5) −	1.5848e+7 (1.99e+5) −	1.7622e+7 (2.71e+4) −	2.0675e+7 (3.51e+4)
WFG8	3	2.7295e+1 (4.85e−1) −	2.8098e+1 (3.27e−1) −	2.7821e+1 (3.24e−1) −	2.7825e+1 (3.80e−1) −	2.8807e+1 (2.59e−1) −	2.9372e+1 (2.48e−1)
	5	1.7367e+3 (1.64e+2) −	3.8951e+3 (3.91e+1) +	3.1389e+3 (7.11e+1) −	3.5199e+3 (5.54e+1) −	3.2468e+3 (7.93e+1) −	3.8241e+3 (4.35e+1)
	8	5.7959e+6 (4.70e+6) −	1.4732e+7 (2.88e+5) −	1.1287e+7 (2.78e+5) −	1.1568e+7 (1.40e+5) −	1.1853e+7 (3.02e+5) −	1.6523e+7 (2.10e+5)
WFG9	3	3.1191e+1 (2.40e+0) −	3.2676e+1 (4.33e−1) −	3.3517e+1 (2.19e−1) −	3.2831e+1 (5.47e−1) −	3.3327e+1 (3.12e−1) −	3.4084e+1 (2.54e−1)
	5	2.3419e+3 (4.40e+2) −	4.1754e+3 (1.73e+2) −	4.1393e+3 (2.07e+2) −	4.2066e+3 (6.28e+1) −	4.1163e+3 (6.78e+1) −	4.6044e+3 (4.08e+1)
	8	1.0250e+7 (4.79e+6) −	1.5390e+7 (7.88e+5) −	1.5401e+7 (4.42e+5) −	1.5036e+7 (3.68e+5) −	1.5800e+7 (3.83e+5) −	1.8621e+7 (1.34e+5)
	-	23	21	24	24	22	
	+	2	1	1	0	3	
	≈	2	5	2	3	2	

Table 4. HV results of six algorithms on benchmarks WFG1-WFG9 with 10 and 15 objectives.

Problems	M	MOEA/D-PaS	NSGA-III	CMOPSO	Two_Arch2	DLEA	DL-TPCEA
WFG1	10	2.4932e+9 (2.20e+8) −	2.9818e+9 (2.95e+8) ≈	2.0618e+9 (3.09e+7) −	2.8738e+9 (2.57e+8) ≈	4.3759e+9 (3.33e+8) +	3.0894e+9 (9.09e+7)
	15	2.8183e+16 (1.31e+15) −	6.2299e+16 (1.10e+16) +	2.8367e+16 (4.81e+14) −	3.6933e+16 (4.35e+15) ≈	7.6448e+16 (6.95e+15) +	3.9888e+16 (6.02e+15)
WFG2	10	3.1504e+9 (1.82e+9) −	9.5411e+9 (6.93e+7) ≈	9.1557e+9 (2.83e+7) −	9.4642e+9 (5.10e+7) −	9.4695e+9 (2.51e+7) −	9.5531e+9 (2.77e+7)
	15	3.5854e+16 (4.24e+16) −	1.7728e+17 (2.08e+14) −	1.5854e+17 (5.06e+15) −	1.7519e+17 (8.15e+14) −	1.7686e+17 (3.04e+14) −	1.7777e+17 (1.70e+14)
WFG3	10	4.9521e−5 (1.99e−7) +	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
	15	6.1526e−16 (1.36e−16) +	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
WFG4	10	3.6386e+9 (3.33e+9) −	8.5986e+9 (9.17e+7) −	6.6569e+10 (8.20e+10) ≈	6.8836e+9 (1.41e+8) −	7.5439e+9 (3.98e+7) −	9.1189e+9 (4.33e+7)
	15	2.9135e+16 (8.76e+15) −	1.5641e+17 (3.18e+15) −	1.1607e+17 (6.92e+15) −	1.2252e+17 (3.15e+15) −	1.4496e+17 (3.45e+15) −	1.7439e+17 (6.93e+14)
WFG5	10	8.0298e+8 (3.32e+7) −	8.2007e+9 (5.81e+7) −	6.4771e+9 (3.09e+8) −	6.3213e+9 (1.22e+8) −	7.1212e+9 (1.27e+8) −	8.5253e+9 (2.97e+7)
	15	1.4143e+16 (4.65e+7) −	1.4816e+17 (2.20e+15) −	1.1418e+17 (3.98e+15) −	1.0054e+17 (2.06e+15) −	1.2722e+17 (4.56e+15) −	1.6240e+17 (3.51e+14)
WFG6	10	2.2633e+9 (1.39e+9) −	7.7778e+9 (1.07e+8) −	5.8760e+9 (2.79e+8) −	6.0002e+9 (1.79e+8) −	6.6823e+9 (1.15e+8) −	8.1978e+9 (2.35e+8)
	15	2.0445e+16 (6.61e+15) −	1.4394e+17 (6.05e+15) −	1.0594e+17 (5.08e+15) −	9.3182e+16 (4.88e+15) −	1.2253e+17 (4.59e+15) −	1.5706e+17 (2.76e+15)
WFG7	10	1.8488e+9 (1.04e+9) −	8.7612e+9 (1.81e+8) −	6.6536e+9 (1.85e+8) −	6.7900e+9 (1.34e+8) −	7.7372e+9 (8.78e+7) −	9.2994e+9 (7.01e+6)
	15	1.6211e+16 (1.17e+14) −	1.5763e+17 (5.67e+15) −	1.2395e+17 (2.42e+15) −	1.0505e+17 (4.37e+15) −	1.4760e+17 (1.91e+15) −	1.7724e+17 (1.34e+14)
WFG8	10	8.6874e+8 (7.58e+6) −	7.5655e+9 (9.99e+7) −	4.4228e+9 (1.09e+8) −	4.8222e+9 (2.53e+8) −	5.3348e+9 (1.81e+8) −	7.8199e+9 (1.26e+8)
	15	1.7958e+16 (2.63e+15) −	1.1199e+17 (1.40e+16) −	6.6492e+16 (1.16e+16) −	7.0410e+16 (1.53e+15) −	1.0321e+17 (4.76e+15) −	1.5825e+17 (3.19e+15)
WFG9	10	1.0134e+9 (7.09e+8) −	7.7585e+9 (2.95e+8) −	6.3588e+9 (2.16e+8) −	6.6188e+9 (1.90e+8) −	6.7972e+9 (1.06e+8) −	8.4373e+9 (7.94e+7)
	15	1.2144e+16 (2.13e+15) −	1.4748e+17 (6.79e+15) ≈	1.1051e+17 (5.64e+15) −	1.0634e+17 (3.16e+15) −	1.2065e+17 (1.51e+15) −	1.5617e+17 (1.04e+16)
	−	16	12	15	14	14	
	+	2	1	0	0	2	
	≈	0	5	3	2	2	

Table 5. IGD results of six algorithms on benchmarks WFG1-WFG9 with 3, 5, and 8 objectives.

Problems	M	MOEA/D-PaS	NSGA-III	CMOPSO	Two_Arch2	DLEA	DL-TPCEA
WFG1	3	1.5489e+0 (2.29e−2) −	1.3950e+0 (5.27e−2) −	1.7560e+0 (7.19e−2) −	1.4184e+0 (4.49e−2) −	1.2451e+0 (8.23e−2) −	1.1443e+0 (7.65e−2)
	5	2.7526e+0 (2.30e−1) −	1.4514e+0 (1.49e−1) ≈	2.2035e+0 (5.35e−2) −	1.7185e+0 (8.59e−2) −	1.4791e+0 (1.27e−1) ≈	1.5296e+0 (1.11e−1)
	8	3.0437e+0 (1.83e−1) −	2.3897e+0 (1.09e−1) ≈	2.8601e+0 (4.19e−2) −	2.3930e+0 (8.43e−2) ≈	2.0988e+0 (1.13e−1) ≈	2.2369e+0 (2.02e−1)
WFG2	3	2.3806e−1 (9.89e−3) −	1.7972e−1 (3.04e−2) −	1.1896e−1 (6.05e−3) −	1.7195e−1 (1.48e−2) −	2.5026e−1 (3.96e−2) −	1.1400e−1 (8.60e−3)
	5	1.7359e+0 (1.66e−1) −	7.1980e−1 (5.74e−2) −	6.0869e−1 (7.43e−2) −	6.2619e−1 (7.94e−2) −	1.3129e+0 (4.11e−1) −	5.1156e−1 (6.59e−2)
	8	6.3124e+0 (3.79e+0) −	3.1921e+0 (1.86e+0) −	1.1504e+0 (4.51e−2) −	1.1816e+0 (1.89e−1) ≈	3.3268e+0 (2.23e−1) −	1.0066e+0 (2.18e−1)
WFG3	3	1.0249e−1 (1.37e−2) ≈	1.6696e−1 (1.62e−2) −	1.2856e−1 (1.03e−2) −	1.5342e−1 (1.84e−2) −	8.6193e−2 (7.43e−3) +	1.1065e−1 (1.13e−2)
	5	1.3530e+0 (1.41e−1) −	5.4731e−1 (1.05e−1) ≈	7.7094e−1 (8.68e−2) −	4.1412e−1 (3.98e−2) +	3.6988e−1 (3.44e−2) +	5.6591e−1 (6.60e−2)
	8	8.8861e+0 (8.66e−5) −	9.8482e−1 (3.15e−1) +	1.5545e+0 (8.06e−2) ≈	9.4384e−1 (3.33e−2) +	7.3548e−1 (8.64e−2) +	1.6935e+0 (3.31e−1)
WFG4	3	2.6154e−1 (9.71e−3) −	1.8267e−1 (3.22e−3) −	2.1399e−1 (4.38e−3) −	1.9309e−1 (5.21e−3) −	2.2205e−1 (7.93e−3) −	1.7016e−1 (3.78e−3)
	5	2.6627e+0 (1.28e−1) −	1.1634e+0 (2.87e−3) −	9.9623e−1 (1.00e−2) −	1.0020e+0 (9.97e−3) −	1.0731e+0 (2.04e−2) −	9.7239e−1 (9.44e−3)
	8	3.4398e+0 (4.68e−2) −	2.7417e+0 (1.38e−2) −	2.6563e+0 (5.37e−2) −	2.8326e+0 (1.04e−2) −	2.7079e+0 (1.13e−2) −	2.5307e+0 (1.51e−2)
WFG5	3	2.1185e−1 (2.89e−3) −	1.9548e−1 (2.95e−3) −	1.8860e−1 (6.74e−3) ≈	2.1768e−1 (4.96e−3) −	2.3407e−1 (8.23e−3) −	1.8664e−1 (3.07e−3)
	5	2.7787e+0 (1.90e−1) −	1.1383e+0 (5.93e−3) −	9.8193e−1 (2.86e−2) +	9.8959e−1 (1.00e−2) ≈	1.0962e+0 (1.72e−2) −	9.9293e−1 (9.04e−3)
	8	3.3155e+0 (3.27e−2) −	2.7859e+0 (1.21e−2) −	2.8877e+0 (8.40e−2) −	2.8167e+0 (2.57e−2) −	2.8605e+0 (4.45e−2) −	2.6346e+0 (3.32e−2)
WFG6	3	3.0029e−1 (3.64e−2) −	2.5494e−1 (1.76e−2) −	1.9560e−1 (7.25e−3) +	2.8551e−1 (1.74e−2) −	2.7869e−1 (1.58e−2) −	2.4262e−1 (1.89e−2)
	5	1.8877e+0 (3.93e−1) −	1.1613e+0 (3.45e−3) −	1.0374e+0 (3.02e−2) ≈	1.0509e+0 (1.60e−2) ≈	1.1386e+0 (2.54e−2) −	1.0488e+0 (1.30e−2)
	8	3.2864e+0 (6.98e−2) −	2.8195e+0 (3.23e−2) ≈	2.9362e+0 (2.23e−2) −	2.9558e+0 (4.11e−2) −	2.9154e+0 (3.73e−2) −	2.8073e+0 (4.48e−2)
WFG7	3	2.2119e−1 (8.24e−3) −	1.8009e−1 (3.35e−3) ≈	1.7571e−1 (3.71e−3) ≈	1.9406e−1 (9.73e−3) −	2.2983e−1 (1.11e−2) −	1.7794e−1 (5.79e−3)
	5	2.9302e+0 (2.47e−1) −	1.1637e+0 (4.06e−3) −	1.0208e+0 (1.51e−2) +	9.8399e−1 (8.08e−3) +	1.1237e+0 (3.00e−2) −	1.0808e+0 (2.16e−2)
	8	4.7379e+0 (3.07e+0) −	2.7927e+0 (1.71e−2) −	2.6952e+0 (2.84e−2) +	2.8153e+0 (3.58e−2) −	2.8150e+0 (3.02e−2) −	2.7405e+0 (2.75e−2)
WFG8	3	3.3340e−1 (1.38e−2) −	3.0618e−1 (6.91e−3) −	3.1422e−1 (8.61e−3) −	3.2821e−1 (1.06e−2) −	3.2300e−1 (9.59e−3) −	2.9003e−1 (5.52e−3)
	5	2.9656e+0 (1.37e−1) −	1.1706e+0 (1.15e−2) −	1.2719e+0 (3.13e−2) −	1.1570e+0 (1.48e−2) −	1.2426e+0 (2.15e−2) −	1.1148e+0 (1.25e−2)
	8	1.1416e+1 (3.81e+0) −	3.1538e+0 (1.14e−2) −	3.3418e+0 (5.20e−2) −	3.4145e+0 (2.80e−2) −	3.2563e+0 (5.28e−2) −	2.9559e+0 (4.04e−2)
WFG9	3	2.5049e−1 (4.26e−2) −	1.8430e−1 (9.41e−3) −	1.6713e−1 (3.72e−3) −	1.8655e−1 (1.19e−2) −	2.0949e−1 (7.53e−3) −	1.6080e−1 (3.23e−3)
	5	2.5243e+0 (1.81e−1) −	1.1054e+0 (1.22e−2) −	9.8803e−1 (3.72e−2) −	9.7439e−1 (1.19e−2) −	1.0773e+0 (2.72e−2) −	9.5847e−1 (1.16e−2)
	8	6.4301e+0 (4.24e+0) −	2.8681e+0 (5.07e−2) −	2.9972e+0 (2.68e−2) −	2.9064e+0 (2.25e−2) −	2.9369e+0 (5.78e−2) −	2.6905e+0 (4.47e−2)
	-	26	21	19	20	22	
	+	0	1	4	3	3	
	≈	1	5	4	4	2	

Table 6. IGD results of six algorithms on benchmarks WFG1-WFG9 with 10 and 15 objectives.

Problems	M	MOEA/D-PaS	NSGA-III	CMOPSO	Two_Arch2	DLEA	DL-TPCEA
WFG1	10	3.4752e+0 (2.18e−1) −	2.5247e+0 (1.47e−1) ≈	3.1837e+0 (3.94e−2) −	2.5480e+0 (8.68e−2) ≈	2.0295e+0 (1.02e−1) +	2.3727e+0 (1.48e−1)
	15	4.4269e+0 (1.80e−1) −	2.8567e+0 (2.36e−1) +	3.9681e+0 (6.38e−2) −	3.5058e+0 (1.05e−1) ≈	2.4277e+0 (1.43e−1) +	3.2972e+0 (2.62e−1)
WFG2	10	8.0643e+0 (1.55e+0) −	5.5784e+0 (3.05e+0) −	1.6248e+0 (1.81e−1) ≈	2.4826e+0 (3.88e−1) −	5.7470e+0 (1.74e+0) −	1.7920e+0 (3.61e−1)
	15	1.7357e+1 (5.54e+0) −	1.1222e+1 (1.89e+0) −	1.5628e+0 (3.78e−1) ≈	1.2642e+0 (2.13e−1) +	1.2789e+1 (1.38e+0) −	2.3050e+0 (7.98e−1)
WFG3	10	1.1189e+1 (1.28e−4) −	9.1908e−1 (6.98e−2) +	2.1021e+0 (1.88e−1) +	1.1879e+0 (8.56e−2) +	1.1060e+0 (1.36e−1) +	2.4613e+0 (2.26e−1)
	15	1.6953e+1 (3.81e−5) −	2.2492e+0 (1.82e−1) +	4.0070e+0 (3.24e−1) ≈	2.1674e+0 (2.62e−1) +	1.6673e+0 (3.86e−1) +	4.2693e+0 (1.23e+0)
WFG4	10	1.3196e+1 (6.43e+0) −	4.4372e+0 (8.31e−2) −	4.7969e+0 (9.82e−1) ≈	4.2424e+0 (5.27e−2) ≈	4.1316e+0 (4.22e−2) ≈	4.2136e+0 (1.04e−1)
	15	2.8206e+1 (9.78e−1) −	8.3714e+0 (4.08e−1) −	7.6517e+0 (8.31e−2) ≈	8.1285e+0 (9.26e−2) −	7.4599e+0 (5.49e−2) +	7.7824e+0 (1.11e−1)
WFG5	10	1.8727e+1 (1.34e−1) −	4.4440e+0 (3.06e−2) −	4.1452e+0 (7.06e−2) −	4.1626e+0 (2.03e−2) −	4.2241e+0 (3.41e−2) −	4.0344e+0 (3.86e−2)
	15	3.0024e+1 (6.03e−8) −	7.8352e+0 (2.05e−1) ≈	7.4354e+0 (4.90e−2) ≈	7.7203e+0 (4.80e−2) −	7.1981e+0 (1.28e−1) +	7.5313e+0 (6.01e−2)
WFG6	10	1.4944e+1 (4.03e+0) −	4.5720e+0 (2.87e−2) ≈	4.2313e+0 (3.21e−2) ≈	4.2511e+0 (3.52e−2) ≈	4.4157e+0 (6.37e−2) ≈	4.3695e+0 (2.07e−1)
	15	2.8561e+1 (2.44e+0) −	1.0013e+1 (1.75e+0) ≈	7.4634e+0 (1.60e−1) +	7.7827e+0 (7.71e−2) ≈	7.3120e+0 (1.30e−1) +	8.0243e+0 (1.90e−1)
WFG7	10	1.6823e+1 (1.73e+0) −	4.5121e+0 (1.30e−1) −	4.0027e+0 (2.17e−2) +	4.1460e+0 (1.77e−2) ≈	4.2551e+0 (5.57e−2) ≈	4.1518e+0 (6.56e−2)
	15	3.0346e+1 (1.61e−3) −	8.8360e+0 (3.39e−1) −	7.2612e+0 (7.48e−2) +	8.2064e+0 (1.16e−1) −	7.1798e+0 (1.01e−1) +	7.6130e+0 (8.70e−2)
WFG8	10	1.9093e+1 (1.36e−1) −	5.1188e+0 (5.07e−1) −	4.6629e+0 (2.17e−2) −	4.8276e+0 (3.38e−2) −	4.6012e+0 (3.56e−2) −	4.2410e+0 (4.91e−2)
	15	2.8903e+1 (1.53e+0) −	9.5721e+0 (7.98e−1) −	7.9078e+0 (6.22e−2) −	8.6835e+0 (1.83e−1) −	7.7107e+0 (6.01e−2) −	7.3664e+0 (2.52e−1)
WFG9	10	1.7540e+1 (2.19e+0) −	4.1996e+0 (7.49e−2) ≈	4.3883e+0 (3.91e−2) −	4.2512e+0 (3.14e−2) −	4.3247e+0 (2.94e−2) −	4.1677e+0 (2.88e−2)
	15	2.9935e+1 (1.05e−1) −	8.1123e+0 (2.73e−1) −	7.7758e+0 (5.19e−2) −	8.0385e+0 (1.01e−1) −	7.7769e+0 (1.58e−1) −	7.2405e+0 (1.43e−1)
	−	18	10	7	9	7	
	+	0	3	4	3	8	
	≈	0	5	7	6	3	

As shown in Tables 3 and 4, in terms of HV, the proposed DL-TPCEA was significantly better than the other five algorithms on 26 out of 45 test instances, and performed similarly to them on two test instances. Specifically, DL-TPCEA generated higher HV values than MOEA/D-PaS, NSGA-III, CMOPSO, Two_Arch2, and DLEA on 39, 33, 39, 38, and 36 out of the 45 test instances, respectively. As shown in Tables 5 and 6, in terms of IGD, the proposed DL-TPCEA was significantly better than the other five algorithms on 21 out of 45 test instances, and performed similarly to them on one test instance. DL-TPCEA generated smaller IGD values than MOEA/D-PaS, NSGA-III, CMOPSO, Two_Arch2, and DLEA on 44, 31, 26, 29, and 29 out of the 45 test instances, respectively. The results demonstrated that it was a promising way to approximate the PFs of WFGs via coevolution and dynamic learning strategy in the proposed DL-TPCEA. CMOPSO and DLEA showed better results for IGD values than for HV values, indicating that these two algorithms also had a good ability to maintain the trade-off between convergence and diversity. In addition, from the comprehensive results of HV values and IGD values, NSGA-III was also a good algorithm. However, compared with these three MaOEAs, the proposed DL-TPCEA also showed much better performance with respect to both convergence and diversity.

The superiority of DL-TPCEA can be explained as follows. The other five comparative algorithms, with the exception of Two_Ach2, attempted to simulate PFs of WFGs through balanced convergence and diversity using a single population. However, as the number of objectives increases, the balance between convergence and diversity became more difficult. This was because the increasing number of objectives led to more serious conflicts on multiple objectives, so that the selection pressure of the MOEAs was not as good as when there were fewer objectives. When the number of objectives kept increasing, the solutions generated by these algorithms may only be single convergence-related solutions or diversity-related solutions, but there was no compromise between convergence and diversity over the whole PF. In addition, Two_Arch2 used two different external archives to store convergence-related solutions or diversity-related solutions, respectively. Moreover, Two_Arch2 promoted the evolution between the two archives so that the population maintained a compromise between convergence and diversity. However, these two external archiving methods had poor performance in dealing with MaOPs, especially the objective conflicts were serious. The proposed DL-TPCEA used two populations for coevolution and the shortcomings of each population will be compensated by BCE. It kept a good balance between convergence and diversity, and used dynamic learning strategy to further strengthen the balance. As a result, the proposed DL-TPCEA did not degrade the performance because of the conflicts caused by the number of objectives increased.

From the HV values in Tables 3 and 4, we can see that the HV value obtained by other related algorithms except MOEA/D-PaS on 8-, 10-, and 15-objective WFG3 was zero. This was caused by the calculation of the HV results using a set of reference points set on the corresponding test instance. When the corresponding algorithms failed to obtain any candidate solution dominating the reference point on those test instances, the value of the hypervolume (HV value) formed by the non-dominated population and the reference point was zero. In these three test instances, only MOEA/D-PaS obtained HV results, which also indicates that MOEA/D-PaS had its own advantages in dealing with WFG3. In addition to the three test instances, all the algorithms obtained HV values (non-zero) on the other test instances.

It can also be concluded from the results that DL-TPCEA mainly showed poor performance on WFG1 and WFG3. In terms of the characteristics of the problem, WFG1 is convex and mixed, while WFG3 is linear and degenerate. DL-TPCEA may not be able to find boundary individuals well on such problems like WFG1 or WFG3, thus, resulting in poor performance. In addition, WFG2 is convex and disconnected. However, DL-TPCEA was still able to obtain the optimal HV results, indicating that the two-population coevolution of DL-TPCEA is capable of dealing with disconnected MaOPs. Finally, WFG4-9 is concave, and DL-TPCEA also has the best performance. The performance of DL-TPCEA on 10- and

15-objective WFG4-9 was lower than DLEA, indicating that dynamic learning strategies played a significant role in dealing with concave MaOPs.

In order to more intuitively observe the ability of the six algorithms to balance convergence and diversity on the WFG test suite, the parallel coordinates of the solution set obtained by the six algorithms on the 5-objective WFG2 and 10-objective WFG9 were given in Figures 3 and 4, respectively. In parallel coordinates, the ordinate represents the objective value, and the convergence information can be obtained. An algorithm has good convergence if it can converge to the range of PF. At the same time, the vertical height can also reflect the performance in the diversity. The horizontal axis corresponds to each objective, which can reflect the diversity information of MaOEAs. It is an algorithm that maintains solutions for every objective, and the denser the lines, the better the diversity. Therefore, using the parallel coordinates of the solution set can better compare the performance of MaOEAs.

For 5-objective WFG2, the range of PF on each objective dimension m is from 0 to $m * 2$ ($m = 1, \ldots, M$). As shown in Figure 3, although the solution sets obtained by all the six algorithms can successfully converged to the range of the corresponding objective dimension on PF, their diversity was significantly different. Among the six algorithms, MOEA/D-PaS and DLEA had the worst performance in diversity. MOEA/D-PaS had a poor diversity in the second objective, while DLEA had a poor diversity in the second to fourth objectives. By contrast, NSGA-III, CMOPSO, and Two_Arch2 algorithms performed better in diversity. However, the diversity of the solution sets obtained by these three algorithms was not as good as that of DL-TPCEA. It can be seen from in Figure 3f that the diversity of the solution set obtained by DL-TPCEA was good in each objective dimension. This indicated that the output solution set of the proposed DL-TPCEA was better than the other five comparative algorithms in terms of convergence and diversity. This result was also consistent with the maximum HV value and minimum IGD value of DL-TPCEA on 5-objective WFG2, as shown in Tables 3 and 5.

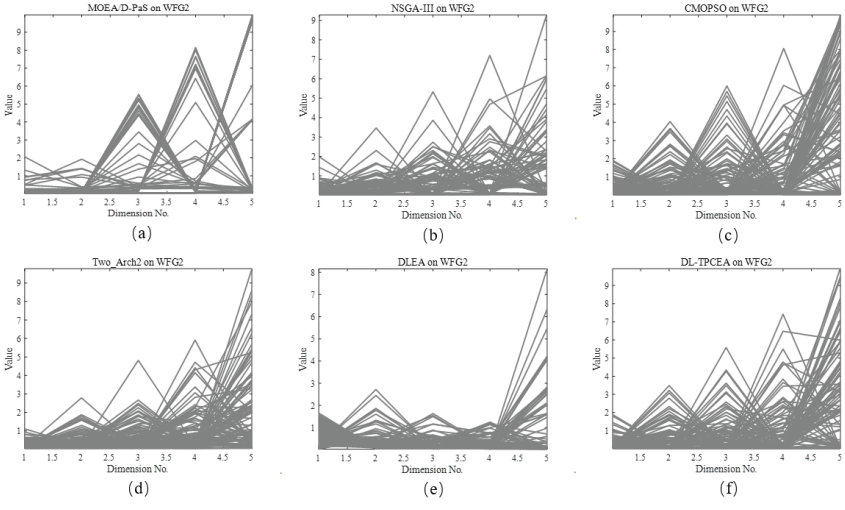


Figure 3. The final solution set obtained by the six MOEAs on 5-objectives WFG2, shown by parallel coordinates. (**a**) The final solution set obtained by MOEA/D-Pas; (**b**) The final solution set obtained by NSGA-III; (**c**) The final solution set obtained by CMOPSO; (**d**) The final solution set obtained by Two_Arch2; (**e**) The final solution set obtained by DLEA; (**f**) The final solution set obtained by DL-TPCEA.

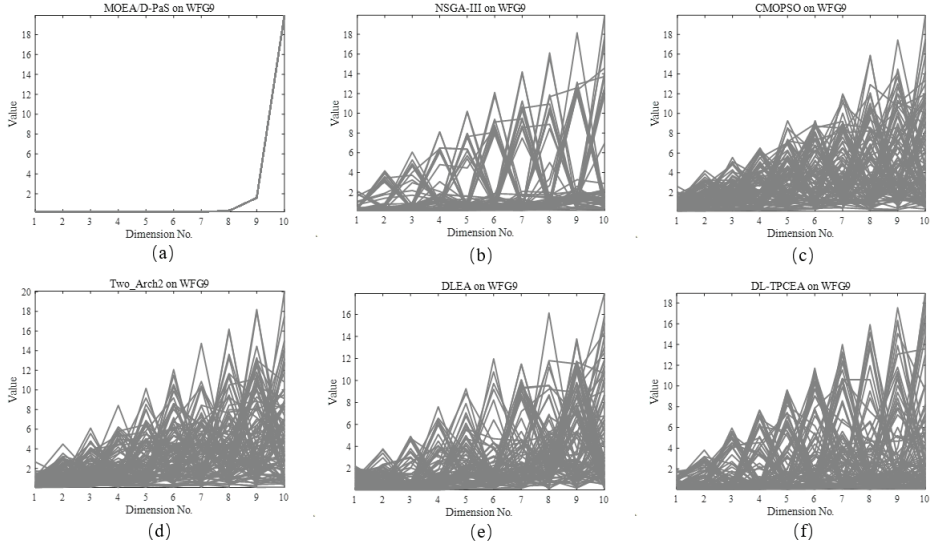


Figure 4. The final solution set obtained by the six MOEAs on 10-objectives WFG9, shown by parallel coordinates. (**a**) The final solution set obtained by MOEA/D-Pas; (**b**) The final solution set obtained by NSGA-III; (**c**) The final solution set obtained by CMOPSO; (**d**) The final solution set obtained by Two_Arch2; (**e**) The final solution set obtained by DLEA; (**f**) The final solution set obtained by DL-TPCEA.

As shown in Figure 4, the solution set obtained by DL-TPCEA was superior to the five comparative algorithms in terms of convergence and diversity. For 10-objective WFG9, the range of PF on each objective dimension m is from 0 to $m * 2$ ($m = 1, \ldots, M$). Among the six algorithms, MOEA/D-PaS converged to few solutions on PF of 10-objective WFG9, so it cannot approach PF well. As shown in Figure 4c,e, the solution set obtained by CMOPSO had a relatively poor diversity on the sixth objective, while DLEA had a relatively poor diversity on the seventh and ninth objectives. NSGA-III and Two_Arch2 algorithms performed well, second only to the convergence and diversity of the solution set obtained by DL-TPCEA on 10-objective WFG9. This was consistent with the HV values and IGD values in Tables 4 and 6.

Figure 5 showed the IGD value trajectories obtained by running six algorithms on the 5-objective WFG test suite. The algorithm for each trajectory was identified in the bottom legend, and DL-TPCEA was specifically highlighted in red. Each subgraph was marked with a different problem, and its abscissa was the number of evaluations during algorithm iteration, and its ordinate was the IGD value. As can be seen from Figure 5, the IGD value trajectory of DL-TPCEA generally declines fastest (except for WFG1 and WFG3), which indicated that DLEA converged very quickly. In addition, from the final result, the IGD value obtained by DL-TPCEA is usually the minimum or not far from the minimum. On 5-objective WFG1, the final IGD value obtained by DL-TPCEA was second only to DLEA. And DLEA obtained the best IGD values on WFG3, while DL-TPCEA performed only at the mid-range level on this issue. Except for WFG1 and WFG3, DL-TPCEA performed very well on the other seven 5-objective WFGs. In general, DL-TPCEA had the best performance among the six algorithms.

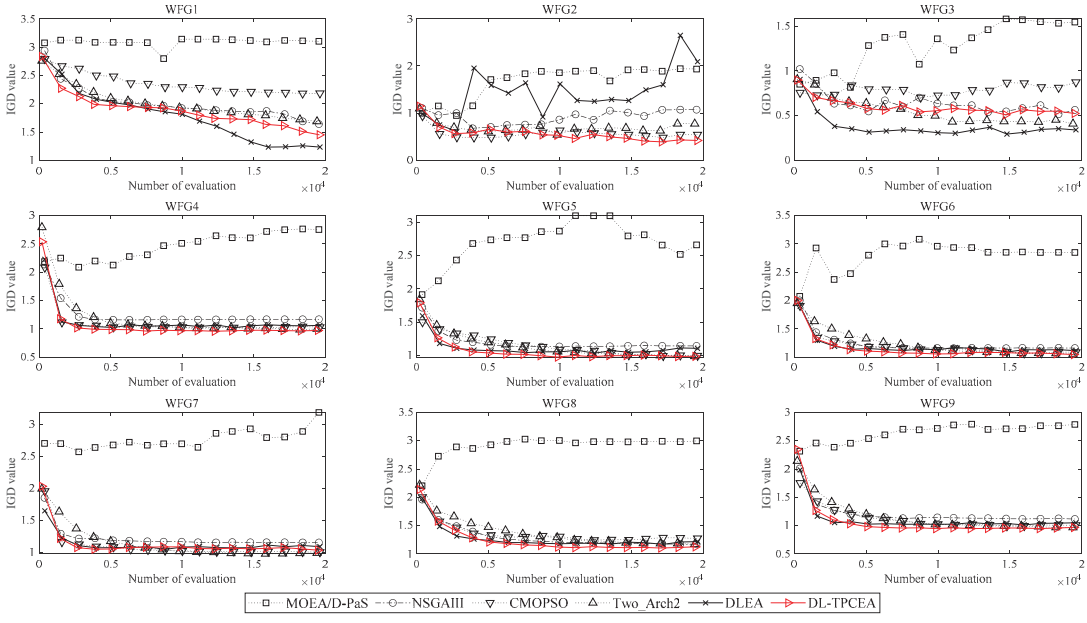


Figure 5. The IGD values convergence trajectories obtained by the six MOEAs on 5-objectives WFG test suite.

Combined with results of HV values and IGD values in Tables 3–6, convergence and diversity effects of solution sets in Figures 3 and 4, as well as IGD value trajectory shown in Figure 5, DL-TPCEA had the best performance in these six algorithms. DL-TPCEA had great advantages in many-objective optimization, both in terms of the convergence and diversity of the final solution set and the convergence speed.

Table 7 shows the average running time of the six comparative algorithms on 3-, 5-, 8-, 10-, and 15-objective WFG1. The last one is the results of Wilcoxon test (the smaller the value is, the shorter the corresponding running time is). The shortest time is NSGA-III, which is due to the simple structure. However, DL-TPCEA is only ranked fourth, which is also a shortcoming. However, given the performance gains, it is worth it, especially for problems that require a lot of accuracy.

Table 7. Comparison of runtime (s) among the six algorithms on 3-, 5-, 8-, 10-, and 15-objective WFG1.

Problem	M	MOEA/D-PaS	NSGA-III	CMOPSO	Two_Arch2	DLEA	DL-TPCEA
WFG1	3	1.7368e+1 (1.75e−1)	6.5904e−1 (3.59e−2)	3.0853e+0 (5.20e−1)	1.1769e+1 (4.49e−1)	1.2042e+1 (8.84e−2)	5.0030e+0 (2.58e−1)
	5	3.5334e+1 (3.27e−1)	1.5763e+0 (6.25e−2)	1.7703e+1 (2.45e+0)	3.8577e+1 (1.06e+0)	2.5697e+1 (1.80e−1)	1.9692e+1 (1.46e+0)
	8	7.2000e+1 (3.02e−1)	2.9553e+0 (5.70e−2)	3.3330e+1 (5.32e+0)	1.0961e+2 (2.00e+0)	6.1095e+1 (5.83e−1)	7.5986e+1 (3.63e+0)
	10	6.7838e+1 (6.60e−1)	3.1321e+0 (2.52e−1)	3.0180e+1 (7.89e+0)	1.1603e+2 (2.26e+0)	5.8485e+1 (2.66e−1)	8.3710e+1 (3.01e+0)
	15	1.0624e+2 (4.48e+0)	4.4098e+0 (1.35e−1)	1.2978e+2 (2.33e+1)	2.4883e+2 (4.10e+0)	1.0903e+2 (7.38e−1)	2.2197e+2 (6.09e+0)
	rank	4.20	1.00	2.40	5.60	3.60	4.20

5.4. Comparison Experiments of DL-TPCEA and Two Weight-Sum Based Algorithms

In this section, we compared DL-TPCEA with two weight-sum based algorithms. The weighted sum method is characterized by fast running speed, simple structure, and easy operation. MaOPs in industrial production tend to have more complex PF, so it may be very limited to solve such problems only by weighted sum method. For this purpose, DL-TPCEA is compared with the weight-sum based approach to verify the advantages of the proposed algorithm.

This paper provides two weight-sum based algorithms for comparison. The first algorithm is a modification of the classic NSGA-II framework, which is called WSEA. The

environment selection of the WSEA starts with a non-dominated sort, and then the rest of the solutions are selected by using weight-sum method in the layer *MaxFNo* mentioned in Section 4.1.2. The environment selection of the second algorithm only selects individuals by weight-sum method, which is called WSEA2. From the point of minimizing the problem, the way to select individuals here is to pick out the N individuals with smallest weighted sum to the next generation. In addition, both algorithms use crossover and mutation operators to generate offspring. Finally, since there is no preference for an objective, the weights of the two algorithms on each objective are set to the same value of $1/M$. However, in order to consider the impact of each objective size on the algorithm, a normalization operation should be carried out for each objective before calculating the weighted sum. DL-TPCEA is compared with the two weight-sum based algorithms mentioned above, and the results are shown in Tables 8–11.

Table 8. HV results of DL-TPCEA and two weight-sum based algorithms on benchmarks WFG1-WFG9 with 3, 5, and 8 objectives.

Problems	M	WSEA	WSEA2	DL-TPCEA
WFG1	3	2.5940e+1 (3.02e+0)	6.9900e+0 (5.88e+0)	2.7839e+1 (1.96e+0)
	5	4.3534e+3 (2.87e+2)	1.5758e+3 (9.81e+2)	2.6535e+3 (2.70e+2)
	8	1.8437e+7 (1.20e+6)	1.1020e+7 (7.58e+6)	6.9852e+6 (1.19e+6)
WFG2	3	4.4022e+1 (1.02e+0)	1.0113e+1 (4.37e+0)	5.8855e+1 (2.83e−1)
	5	4.2104e+3 (9.81e+2)	2.2314e+3 (1.04e+3)	6.1122e+3 (1.83e+1)
	8	1.6107e+7 (2.25e+6)	1.6424e+7 (2.47e+6)	2.1889e+7 (8.02e+4)
WFG3	3	4.7170e+0 (3.09e−1)	1.4324e+0 (1.07e−2)	5.9102e+0 (9.30e−2)
	5	1.2563e+0 (9.19e−2)	1.0850e+0 (6.56e−3)	1.1142e+0 (3.37e−1)
	8	1.6284e−2 (2.65e−3)	1.3718e−2 (1.29e−3)	0.0000e+0 (0.00e+0)
WFG4	3	2.1540e+1 (2.61e−1)	5.8596e+0 (1.05e−1)	3.5106e+1 (1.47e−1)
	5	1.5536e+3 (1.18e+2)	7.4547e+2 (1.23e+2)	4.8905e+3 (2.30e+1)
	8	5.1626e+6 (1.10e+6)	4.0105e+6 (8.00e+5)	2.0259e+7 (1.20e+5)
WFG5	3	5.6176e+0 (3.55e−1)	4.9062e+0 (2.20e−4)	3.3072e+1 (2.45e−1)
	5	4.8517e+2 (7.18e−4)	4.8516e+2 (2.96e−6)	4.5798e+3 (1.63e+1)
	8	1.7489e+6 (4.13e+0)	1.7489e+6 (4.42e+0)	1.8993e+7 (7.80e+4)
WFG6	3	1.9143e+1 (8.51e−1)	5.2439e+0 (9.21e−1)	3.1280e+1 (8.81e−1)
	5	1.4861e+3 (4.26e+2)	8.3328e+2 (4.21e+2)	4.3856e+3 (8.73e+1)
	8	4.0300e+6 (1.18e+6)	2.3463e+6 (1.30e+5)	1.8534e+7 (4.34e+5)
WFG7	3	1.7538e+1 (1.22e+0)	9.4479e+0 (3.78e+0)	3.5763e+1 (9.24e−2)
	5	1.5602e+3 (3.41e+2)	8.5188e+2 (3.01e+2)	4.9534e+3 (8.29e+0)
	8	6.0797e+6 (7.26e+5)	4.7256e+6 (1.71e+6)	2.0675e+7 (3.51e+4)
WFG8	3	1.6581e+1 (2.50e+0)	5.6755e+0 (1.60e−1)	2.9372e+1 (2.48e−1)
	5	6.8821e+2 (6.54e+1)	6.1305e+2 (7.08e+1)	3.8241e+3 (4.35e+1)
	8	2.2857e+6 (6.42e+5)	2.0356e+6 (3.05e+5)	1.6523e+7 (2.10e+5)
WFG9	3	8.3332e+0 (1.99e+0)	3.4207e+0 (4.40e−5)	3.4084e+1 (2.54e−1)
	5	6.5479e+2 (9.07e+1)	4.4075e+2 (9.90e+1)	4.6044e+3 (4.08e+1)
	8	3.0759e+6 (7.92e+5)	2.8697e+6 (5.56e+5)	1.8621e+7 (1.34e+5)

Table 9. HV results of DL-TPCEA and two weight-sum based algorithms on benchmarks WFG1-WFG9 with 10 and 15 objectives.

Problems	M	WSEA	WSEA2	DL-TPCEA
WFG1	10	8.3451e+9 (3.40e+8)	8.0823e+9 (8.38e+8)	3.0894e+9 (9.09e+7)
	15	1.3885e+17 (1.57e+14)	1.3151e+17 (7.93e+15)	3.9888e+16 (6.02e+15)
WFG2	10	7.7232e+9 (5.92e+8)	5.1805e+9 (1.83e+9)	9.5531e+9 (2.77e+7)
	15	1.2874e+17 (2.49e+16)	1.0902e+17 (2.21e+16)	1.7777e+17 (1.70e+14)
WFG3	10	3.8860e−5 (5.32e−6)	3.8105e−5 (5.43e−6)	0.0000e+0 (0.00e+0)
	15	0.0000e+0 (0.00e+0)	0.0000e+0 (0.00e+0)	0.0000e+0 (0.00e+0)
WFG4	10	2.0785e+9 (6.77e+8)	2.5030e+9 (7.61e+8)	9.1189e+9 (4.33e+7)
	15	4.9360e+16 (2.12e+16)	3.1410e+16 (1.15e+16)	1.7439e+17 (6.93e+14)
WFG5	10	7.6254e+8 (9.65e−1)	7.6254e+8 (4.37e+1)	8.5253e+9 (2.97e+7)
	15	1.4147e+16 (1.75e+12)	1.4146e+16 (1.83e+12)	1.6240e+17 (3.51e+14)
WFG6	10	1.8340e+9 (5.01e+8)	1.7364e+9 (7.30e+8)	8.1978e+9 (2.35e+8)
	15	4.5548e+16 (1.85e+16)	3.2527e+16 (1.02e+16)	1.5706e+17 (2.76e+15)
WFG7	10	2.8722e+9 (6.09e+8)	2.2976e+9 (9.57e+8)	9.2994e+9 (7.01e+6)
	15	4.9910e+16 (1.06e+16)	3.4728e+16 (9.38e+15)	1.7724e+17 (1.34e+14)
WFG8	10	8.7211e+8 (9.80e+7)	8.6477e+8 (2.00e+8)	7.8199e+9 (1.26e+8)
	15	3.6695e+16 (2.82e+16)	4.2377e+16 (9.52e+15)	1.5825e+17 (3.19e+15)
WFG9	10	1.6703e+9 (9.25e+8)	1.4451e+9 (5.64e+8)	8.4373e+9 (7.94e+7)
	15	5.4094e+16 (7.47e+15)	4.6128e+16 (2.02e+16)	1.5617e+17 (1.04e+16)

As can be seen from the results in Tables 8–11, DL-TPCEA obtained the optimal results in all the other instances except for the HV results of seven instances on WFG1 and WFG3. Regardless of HV or IGD indicator, DL-TPCEA has the best performance among the three algorithms. It also reflects that the complexity of MaOPs cannot be well adapted to only relying on a single weighted sum method. The reasons are as follows: without considering the objective preference, it is not guaranteed that the solution in the population will converge to PF only by the magnitude of the weighted sum. A smaller weighted sum may just be that the individual retains a smaller objective value for some objective, but whether the individual is a non-dominated solution is unknown. In addition, the method based on the weighted sum is linearly convergent. However, different MaOPs have different characteristics, making it difficult to apply this method to all problems.

From the point of the feature of the problem, WFG1 is convex and mixed, while WFG3 is linear and degenerate. WSEA has just obtained the optimal HV results in several examples of these two problems, indicating the weighted sum method is promising to deal with these problems. However, the IGD values obtained by WSEA and WSEA2 on these examples are very poor, which also indicates that the convergence ability is not strong. The calculation of HV will consider some boundary individuals in the population, so DL-TPCEA may not get good HV results because of the boundary individuals in the population. However, combining the results of the two indicators, DL-TPCEA performed best in 83 out of 90 instances, which is an overwhelming advantage. These results reflect the limitations of using the weighted sum method to solve MaOPs, and show that DL-TPCEA has good advantages.

Table 10. IGD results of DL-TPCEA and two weight-sum based algorithms on benchmarks WFG1-WFG9 with 3, 5, and 8 objectives.

Problems	M	WSEA	WSEA2	DL-TPCEA
WFG1	3	1.2881e+0 (1.35e−1)	3.4165e+0 (9.29e−1)	1.1443e+0 (7.65e−2)
	5	2.0015e+0 (1.18e−1)	5.0632e+0 (1.85e+0)	1.5296e+0 (1.11e−1)
	8	2.4999e+0 (2.12e−1)	7.6394e+0 (4.79e+0)	2.2369e+0 (2.02e−1)
WFG2	3	7.7414e−1 (4.59e−2)	3.1113e+0 (3.99e−1)	1.1400e−1 (8.60e−3)
	5	1.6495e+0 (4.13e−1)	2.7955e+0 (9.76e−1)	5.1156e−1 (6.59e−2)
	8	3.8914e+0 (6.35e−1)	3.5795e+0 (5.84e−1)	1.0066e+0 (2.18e−1)
WFG3	3	1.4793e+0 (1.96e−1)	3.2014e+0 (2.81e−3)	1.1065e−1 (1.13e−2)
	5	5.2586e+0 (1.01e−1)	5.4443e+0 (4.88e−4)	5.6591e−1 (6.60e−2)
	8	8.5239e+0 (3.90e−1)	8.8671e+0 (3.27e−2)	1.6935e+0 (3.31e−1)
WFG4	3	1.4651e+0 (1.51e−2)	3.9049e+0 (7.37e−2)	1.7016e−1 (3.78e−3)
	5	5.7459e+0 (2.93e−1)	7.4360e+0 (3.14e−1)	9.7239e−1 (9.44e−3)
	8	1.1621e+1 (1.03e+0)	1.2737e+1 (4.56e−1)	2.5307e+0 (1.51e−2)
WFG5	3	3.6430e+0 (9.53e−2)	3.8534e+0 (6.71e−5)	1.8664e−1 (3.07e−3)
	5	7.9990e+0 (5.15e−6)	7.9990e+0 (2.13e−8)	9.9293e−1 (9.04e−3)
	8	1.4670e+1 (9.68e−6)	1.4670e+1 (1.06e−5)	2.6346e+0 (3.32e−2)
WFG6	3	1.5039e+0 (7.48e−2)	3.7653e+0 (2.26e−1)	2.4262e−1 (1.89e−2)
	5	5.3476e+0 (8.43e−1)	7.0610e+0 (9.93e−1)	1.0488e+0 (1.30e−2)
	8	1.2326e+1 (1.24e+0)	1.3945e+1 (1.32e−1)	2.8073e+0 (4.48e−2)
WFG7	3	1.8260e+0 (1.95e−1)	3.1263e+0 (6.97e−1)	1.7794e−1 (5.79e−3)
	5	5.1307e+0 (5.05e−1)	7.2260e+0 (7.30e−1)	1.0808e+0 (2.16e−2)
	8	1.1043e+1 (4.85e−1)	1.1873e+1 (1.83e+0)	2.7405e+0 (2.75e−2)
WFG8	3	1.5861e+0 (3.63e−1)	3.8409e+0 (2.16e−1)	2.9003e−1 (5.52e−3)
	5	5.5266e+0 (1.98e−1)	6.6281e+0 (8.22e−1)	1.1148e+0 (1.25e−2)
	8	1.1010e+1 (1.07e+0)	1.3450e+1 (1.43e+0)	2.9559e+0 (4.04e−2)
WFG9	3	3.1525e+0 (2.30e−1)	3.8736e+0 (1.04e−6)	1.6080e−1 (3.23e−3)
	5	7.5318e+0 (2.87e−1)	7.9566e+0 (6.88e−2)	9.5847e−1 (1.16e−2)
	8	1.3446e+1 (6.85e−1)	1.3468e+1 (5.75e−1)	2.6905e+0 (4.47e−2)

Table 11. IGD results of DL-TPCEA and two weight-sum based algorithms on benchmarks WFG1-WFG9 with 10 and 15 objectives.

Problems	M	WSEA	WSEA2	DL-TPCEA
WFG1	10	2.7506e+0 (4.24e−1)	3.2357e+0 (1.43e+0)	2.3727e+0 (1.48e−1)
	15	3.3141e+0 (1.74e−1)	3.5682e+0 (7.23e−1)	3.2972e+0 (2.62e−1)
WFG2	10	5.1728e+0 (6.06e−1)	6.3742e+0 (1.88e+0)	1.7920e+0 (3.61e−1)
	15	1.2806e+1 (1.47e+0)	1.3168e+1 (2.04e+0)	2.3050e+0 (7.98e−1)
WFG3	10	1.1169e+1 (2.14e−2)	1.1187e+1 (2.10e−3)	2.4613e+0 (2.26e−1)
	15	1.5948e+1 (1.23e+0)	1.6847e+1 (1.76e−1)	4.2693e+0 (1.23e+0)

Table 11. *Cont.*

Problems	M	WSEA	WSEA2	DL-TPCEA
WFG4	10	1.5880e+1 (1.63e+0)	1.5367e+1 (1.51e+0)	4.2136e+0 (1.04e−1)
	15	2.5245e+1 (2.83e+0)	2.7162e+1 (1.53e+0)	7.7824e+0 (1.11e−1)
WFG5	10	1.8913e+1 (6.20e−9)	1.8913e+1 (6.95e−8)	4.0344e+0 (3.86e−2)
	15	3.0022e+1 (6.26e−4)	3.0023e+1 (6.52e−4)	7.5313e+0 (6.01e−2)
WFG6	10	1.5554e+1 (1.64e+0)	1.6302e+1 (1.94e+0)	4.3695e+0 (2.07e−1)
	15	2.4092e+1 (2.70e+0)	2.5533e+1 (2.47e+0)	8.0243e+0 (1.90e−1)
WFG7	10	1.3849e+1 (1.58e+0)	1.4887e+1 (2.41e+0)	4.1518e+0 (6.56e−2)
	15	2.2743e+1 (2.69e+0)	2.6167e+1 (2.09e+0)	7.6130e+0 (8.70e−2)
WFG8	10	1.4975e+1 (1.68e+0)	1.5039e+1 (1.52e+0)	4.2410e+0 (4.91e−2)
	15	2.4219e+1 (4.49e+0)	2.1602e+1 (2.28e+0)	7.3664e+0 (2.52e−1)
WFG9	10	1.6810e+1 (2.26e+0)	1.7056e+1 (1.58e+0)	4.1677e+0 (2.88e−2)
	15	2.3731e+1 (1.25e+0)	2.5088e+1 (3.06e+0)	7.2405e+0 (1.43e−1)

6. Conclusions

In recent years, in order to enable MOEAs to handle MaOPs with various characteristics, various MOEAs have been proposed. However, these MOEAs also had their own disadvantages. For example, MOEAs that rely on reference vectors cannot well represent the characteristics of the whole PF when generating reference vectors, which results in the performance degradation of MOEAs. This paper made full use of the advantages of DLS in many-objective optimization (better to maintain convergence and diversity), and proposed DL-TPCEA in combination with the BCE framework. The effective combination of the two strategies can further explore the entire decision space. At the same time, the convergence factor in DLS is further improved according to the evolutionary state of the population in BCE, and then the dynamic convergence factor is proposed to better use the important element of the evolutionary state of the population. This effective combination greatly improves the performance of DL-TPCEA. When compared with five state-of-the-art MOEAs, DL-TPCEA has significant advantages. Finally, in order to verify the performance advantage of DL-TPCEA over the weight-sum based algorithm, DL-TPCEA was compared with the two weight-sum based algorithms, and the results showed that DL-TPCEA still had significant advantages.

In addition, the original DLS used $I_{\varepsilon+}$ to maintain individual convergence and a diversity maintenance mechanism based on L_p-norm distance to maintain diversity. In this paper, the CV indicator is used to maintain individual convergence, and the comparison between CV and $I_{\varepsilon+}$ should be the future research direction. In addition, there are still many excellent strategies that can be used to maintain convergence and diversity, and this paper does not compare these strategies. The future direction of work can start from this point and be improved under the framework of DL-TPCEA to achieve better results. We used dynamic learning factors to combine DLS and BCE more effectively, but there are more ways to combine them more effectively in the future. In terms of the selection of the initial value of the dynamic convergence factor, suggestions in relevant paper [94] can also be referred to get a better initial value.

Author Contributions: Conceptualization, G.L. and G.-G.W.; methodology, G.L.; software, G.L.; validation, G.-G.W. and S.W.; formal analysis, S.W.; investigation, G.L.; resources, G.L. and G.-G.W.; data curation, G.L.; writing—original draft preparation, G.L.; writing—review and editing, G.-G.W. and S.W.; visualization, G.L.; supervision, G.L., G.-G.W. and S.W.; project administration, G.-G.W. and S.W.; funding acquisition, G.-G.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China, grant number U1706218, 41576011, 41706010 and 61503165.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chen, H.; Cheng, R.; Wen, J.; Li, H.; Weng, J. Solving large-scale many-objective optimization problems by covariance matrix adaptation evolution strategy with scalable small subpopulations. *Inf. Sci.* **2020**, *509*, 457–469. [CrossRef]
2. Yang, S.; Li, M.; Liu, X.; Zheng, J. A grid-based evolutionary algorithm for many-objective optimization. *IEEE Trans. Evol. Comput.* **2013**, *17*, 721–736. [CrossRef]
3. Ishibuchi, H.; Tsukamoto, N.; Nojima, Y. Evolutionary many-objective optimization: A short review. In Proceedings of the 2008 IEEE Congress on Evolutionary Computation, Hong Kong, China, 1–6 June 2008; pp. 2419–2426.
4. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
5. Deb, K.; Jain, H. An evolutionary many-objective optimization slgorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Trans. Evol. Comput.* **2014**, *18*, 577–601. [CrossRef]
6. Jain, H.; Deb, K. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach. *IEEE Trans. Evol. Comput.* **2014**, *18*, 602–622. [CrossRef]
7. Zhang, X.; Tian, Y.; Jin, Y. A knee point-driven evolutionary algorithm for many-objective optimization. *IEEE Trans. Evol. Comput.* **2015**, *19*, 761–776. [CrossRef]
8. Zitzler, E.; Künzli, S. Indicator-based selection in multiobjective search. In Proceedings of the International conference on parallel problem solving from nature, Birmingham, UK, 18–22 September 2004; pp. 832–842.
9. Bader, J.; Zitzler, E. HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evol. Comput.* **2011**, *19*, 45–76. [CrossRef]
10. Zitzler, E.; Thiele, L.; Laumanns, M.; Fonseca, C.M.; da Fonseca, V.G. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Trans. Evol. Comput.* **2003**, *7*, 117–132. [CrossRef]
11. While, L.; Hingston, P.; Barone, L.; Huband, S. A faster algorithm for calculating hypervolume. *IEEE Trans. Evol. Comput.* **2006**, *10*, 29–38. [CrossRef]
12. Russo, L.M.S.; Francisco, A.P. Quick hypervolume. *IEEE Trans. Evol. Comput.* **2014**, *18*, 481–502. [CrossRef]
13. Zhang, Q.; Li, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **2007**, *11*, 712–731. [CrossRef]
14. Wang, R.; Zhang, Q.; Zhang, T. Decomposition-based algorithms using Pareto adaptive scalarizing methods. *IEEE Trans. Evol. Comput.* **2016**, *20*, 821–837. [CrossRef]
15. Potter, M.A.; Jong, K.A.D. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evol. Comput.* **2000**, *8*, 1–29. [CrossRef] [PubMed]
16. Wiegand, R.P.; Potter, M.A. Robustness in cooperative coevolution. In Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, Seattle, WA, USA, 8–12 July 2006; pp. 369–376.
17. Wiegand, R.P.; Liles, W.C.; de Jong, K.A. Analyzing cooperative coevolution with evolutionary game theory. In Proceedings of the 2002 IEEE Congress on Evolutionary Computation, Honolulu, HI, USA, 12–17 May 2002; pp. 1600–1605.
18. De Jong, K.A.; Potter, M.A. Evolving complex structures via cooperative coevolution. In *Evolutionary Programming IV*; McDonnell, J.R., Reynolds, R.G., Fogel, D.B., Eds.; The MIT Press: Cambridge, MA, USA, 1995; pp. 307–317.
19. Sofge, D.; de Jong, K.; Schultz, A. A blended population approach to cooperative coevolution for decomposition of complex problems. In Proceedings of the 2002 IEEE Congress on Evolutionary Computation, Honolulu, HI, USA, 12–17 May 2002; pp. 413–418.
20. Yang, Z.; Tang, K.; Yao, X. Large scale evolutionary optimization using cooperative coevolution. *Inf. Sci.* **2008**, *178*, 2985–2999. [CrossRef]
21. Yang, Z.; Tang, K.; Yao, X. Multilevel cooperative coevolution for large scale optimization. In Proceedings of the 2008 IEEE Congress on Evolutionary Computation, Hong Kong, China, 1–6 June 2008; pp. 1663–1670.
22. Liu, Y.; Yao, X.; Zhao, Q.; Higuchi, T. Scaling up fast evolutionary programming with cooperative coevolution. In Proceedings of the 2001 IEEE Congress on Evolutionary Computation, Seoul, South Korea, 27–30 May 2001; pp. 1101–1108.
23. Bucci, A.; Pollack, J.B. On identifying global optima in cooperative coevolution. In Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, Washington, DC, USA, 25–29 June 2005; pp. 539–544.
24. Chen, W.; Weise, T.; Yang, Z.; Tang, K. Large-scale global optimization using cooperative coevolution with variable interaction learning. In Proceedings of the International Conference on Parallel Problem Solving from Nature, Kraków, Poland, 11–15 September 2010; pp. 300–309.
25. Tan, T.G.; Teo, J.; Lau, H.K. Performance scalability of a cooperative coevolution multiobjective evolutionary algorithm. In Proceedings of the 2007 IEEE International Conference on Computational Intelligence and Security, Washington, DC, USA, 15–19 December 2007; pp. 119–123.
26. Wang, R.; Purshouse, R.C.; Fleming, P.J. Preference-inspired coevolutionary algorithms for many-objective optimization. *IEEE Trans. Evol. Comput.* **2012**, *17*, 474–494. [CrossRef]

27. Purshouse, R.C.; Jalbă, C.; Fleming, P.J. Preference-driven co-evolutionary algorithms show promise for many-objective optimisation. In Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization, Ouro Preto, Brazil, 5–8 April 2011; pp. 136–150.
28. Wang, R.; Purshouse, R.C.; Fleming, P.J. On finding well-spread Pareto optimal solutions by preference-inspired co-evolutionary algorithm. In Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, Amsterdam, The Netherlands, 6–10 July 2013; pp. 695–702.
29. Wang, R.; Purshouse, R.C.; Fleming, P.J. Preference-inspired co-evolutionary algorithms using weight vectors. *Eur. J. Oper. Res.* **2015**, *243*, 423–441. [CrossRef]
30. Liang, Z.; Wang, X.; Lin, Q.; Chen, F.; Chen, J.; Ming, Z. A novel multi-objective co-evolutionary algorithm based on decomposition approach. *Appl. Soft Comput.* **2018**, *73*, 50–66. [CrossRef]
31. Zhang, Y.-H.; Gong, Y.-J.; Gu, T.-L.; Yuan, H.-Q.; Zhang, W.; Kwong, S.; Zhang, J. DECAL: Decomposition-based coevolutionary algorithm for many-objective optimization. *IEEE Trans. Cybern.* **2017**, *49*, 27–41. [CrossRef] [PubMed]
32. Shu, Z.; Wang, W. Preference-inspired co-evolutionary algorithms with local PCA oriented goal vectors for many-objective optimization. *IEEE Access* **2018**, *6*, 68701–68715. [CrossRef]
33. Antonio, L.M.; Coello, C.A.C. Use of cooperative coevolution for solving large scale multiobjective optimization problems. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; pp. 2758–2765.
34. Gong, D.; Xu, B.; Zhang, Y.; Guo, Y.; Yang, S. A similarity-based cooperative co-evolutionary algorithm for dynamic interval multi-objective optimization problems. *IEEE Trans. Evol. Comput.* **2019**, *24*, 142–156. [CrossRef]
35. Sun, J.; Miao, Z.; Gong, D.; Zeng, X.; Li, J.; Wang, G. Interval multiobjective optimization with memetic algorithms. *IEEE Trans. Cybern.* **2020**, *50*, 3444–3457. [CrossRef]
36. Yi, J.-H.; Xing, L.-N.; Wang, G.-G.; Dong, J.; Vasilakos, A.V.; Alavi, A.H.; Wang, L. Behavior of crossover operators in NSGA-III for large-scale optimization problems. *Inf. Sci.* **2020**, *509*, 470–487. [CrossRef]
37. González, J.; Ortega, J.; Damas, M.; Martín-Smith, P. Many-objective cooperative co-evolutionary feature selection: A lexicographic approach. In Proceedings of the International Work-Conference on Artificial Neural Networks, Gran Canaria, Spain, 12–14 June 2019; pp. 463–474.
38. Wang, F.; Li, Y.; Liao, F.; Yan, H. An ensemble learning based prediction strategy for dynamic multi-objective optimization. *Appl. Soft Comput.* **2020**, *96*, 106592. [CrossRef]
39. Wang, F.; Li, Y.; Zhang, H.; Hu, T.; Shen, X.-L. An adaptive weight vector guided evolutionary algorithm for preference-based multi-objective optimization. *Swarm Evol. Comput.* **2019**, *49*, 220–233. [CrossRef]
40. Shen, X.; Guo, Y.; Li, A. Cooperative coevolution with an improved resource allocation for large-scale multi-objective software project scheduling. *Appl. Soft Comput.* **2020**, *88*, 106059. [CrossRef]
41. Liu, X.; Zhan, Z.; Gao, Y.; Zhang, J.; Kwong, S.; Zhang, J. Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization. *IEEE Trans. Evol. Comput.* **2019**, *23*, 587–602. [CrossRef]
42. Zhou, Y.; Yang, J.; Zheng, L. Hyper-heuristic coevolution of machine assignment and job sequencing rules for multi-objective dynamic flexible job shop scheduling. *IEEE Access* **2019**, *7*, 68–88. [CrossRef]
43. Li, M.; Yang, S.; Liu, X. Pareto or non-Pareto: Bi-criterion evolution in multiobjective optimization. *IEEE Trans. Evol. Comput.* **2015**, *20*, 645–665. [CrossRef]
44. Yuan, J.; Liu, H.-L.; Gu, F. A cost value based evolutionary many-objective optimization algorithm with neighbor selection strategy. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation, Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
45. Sang, H.-Y.; Pan, Q.-K.; Li, J.-Q.; Wang, P.; Han, Y.-Y.; Gao, K.-Z.; Duan, P. Effective invasive weed optimization algorithms for distributed assembly permutation flowshop problem with total flowtime criterion. *Swarm Evol. Comput.* **2019**, *44*, 64–73. [CrossRef]
46. Gao, D.; Wang, G.G.; Pedrycz, W. Solving fuzzy job-shop scheduling problem using DE algorithm improved by a selection mechanism. *IEEE Trans. Fuzzy Syst.* **2020**, *28*, 3265–3275. [CrossRef]
47. Li, J.-Q.; Pan, Q.-K.; Tasgetiren, M.F. A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities. *Appl. Math. Model.* **2014**, *38*, 1111–1132. [CrossRef]
48. Han, Y.-Y.; Gong, D.; Sun, X. A discrete artificial bee colony algorithm incorporating differential evolution for the flow-shop scheduling problem with blocking. *Eng. Optimiz.* **2015**, *47*, 927–946. [CrossRef]
49. Sang, H.-Y.; Pan, Q.-K.; Duan, P.-Y.; Li, J.-Q. An effective discrete invasive weed optimization algorithm for lot-streaming flowshop scheduling problems. *J. Intell. Manuf.* **2018**, *29*, 1337–1349. [CrossRef]
50. Wang, G.-G.; Gandomi, A.H.; Zhao, X.; Chu, H.C. Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. *Soft Comput.* **2016**, *20*, 273–285. [CrossRef]
51. Zhao, X.-C. A perturbed particle swarm algorithm for numerical optimization. *Appl. Soft Comput.* **2010**, *10*, 119–124.
52. Wang, H.; Yi, J.-H. An improved optimization method based on krill herd and artificial bee colony with information exchange. *Memetic Comput.* **2018**, *10*, 177–198. [CrossRef]
53. Liu, F.; Sun, Y.; Wang, G.-G.; Wu, T. An artificial bee colony algorithm based on dynamic penalty and Lévy flight for constrained optimization problems. *Arab. J. Sci. Eng.* **2018**, *43*, 7189–7208. [CrossRef]
54. Zhang, Y.; Gong, D.; Hu, Y.; Zhang, W. Feature selection algorithm based on bare bones particle swarm optimization. *Neurocomputing* **2015**, *148*, 150–157. [CrossRef]
55. Wang, G.-G.; Deb, S.; Cui, Z. Monarch butterfly optimization. *Neural Comput. Appl.* **2019**, *31*, 1995–2014. [CrossRef]

56. Wang, G.-G.; Deb, S.; Zhao, X.; Cui, Z. A new monarch butterfly optimization with an improved crossover operator. *Oper. Res.* **2018**, *18*, 731–755. [CrossRef]
57. Feng, Y.; Wang, G.-G.; Li, W.; Li, N. Multi-strategy monarch butterfly optimization algorithm for discounted {0-1} knapsack problem. *Neural Comput. Appl.* **2018**, *30*, 3019–3036. [CrossRef]
58. Feng, Y.; Deb, S.; Wang, G.-G.; Alavi, A.H. Monarch butterfly optimization: A comprehensive review. *Expert Syst. Appl.* **2021**, *168*, 114418. [CrossRef]
59. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [CrossRef]
60. Zhao, D.; Liu, L.; Yu, F.; Heidari, A.A.; Wang, M.; Liang, G.; Muhammad, K.; Chen, H. Chaotic random spare ant colony optimization for multi-threshold image segmentation of 2D Kapur entropy. *Knowl. Based Syst.* **2021**, *216*, 106510. [CrossRef]
61. Gandomi, A.H.; Alavi, A.H. Krill herd: A new bio-inspired optimization algorithm. *Commun. Nonlinear Sci. Numer. Simulat.* **2012**, *17*, 4831–4845. [CrossRef]
62. Wang, G.-G.; Gandomi, A.H.; Alavi, A.H. An effective krill herd algorithm with migration operator in biogeography-based optimization. *Appl. Math. Model.* **2014**, *38*, 2454–2462. [CrossRef]
63. Wang, G.-G.; Gandomi, A.H.; Alavi, A.H.; Gong, D. A comprehensive review of krill herd algorithm: Variants, hybrids and applications. *Artif. Intell. Rev.* **2019**, *51*, 119–148. [CrossRef]
64. Wang, G.-G.; Guo, L.; Gandomi, A.H.; Hao, G.-S.; Wang, H. Chaotic krill herd algorithm. *Inf. Sci.* **2014**, *274*, 17–34. [CrossRef]
65. Li, W.; Wang, G.-G.; Alavi, A.H. Learning-based elephant herding optimization algorithm for solving numerical optimization problems. *Knowl. Based Syst.* **2020**, *195*, 105675. [CrossRef]
66. Li, J.; Lei, H.; Alavi, A.H.; Wang, G.-G. Elephant Herding Optimization: Variants, Hybrids, and Applications. *Mathematics* **2020**, *8*, 1415. [CrossRef]
67. Li, W.; Wang, G.-G.; Gandomi, A.H. A Survey of learning-based intelligent optimization algorithms. *Arch. Comput. Method. E* **2021**, in press. [CrossRef]
68. Wang, G.-G.; Tan, Y. Improving metaheuristic algorithms with information feedback models. *IEEE Trans. Cybern.* **2019**, *49*, 542–555. [CrossRef] [PubMed]
69. Wang, F.; Li, Y.; Zhou, A.; Tang, K. An estimation of distribution algorithm for mixed-variable newsvendor problems. *IEEE Trans. Evol. Comput.* **2020**, *24*, 479–493. [CrossRef]
70. Rizk-Allah, R.M.; El-Sehiemy, R.A.; Deb, S.; Wang, G.-G. A novel fruit fly framework for multi-objective shape design of tubular linear synchronous motor. *J. Supercomput.* **2017**, *73*, 1235–1256. [CrossRef]
71. Hemmelmayr, V.C.; Cordeau, J.-F.; Crainic, T.G. An adaptive large neighborhood search heuristic for Two-Echelon Vehicle Routing Problems arising in city logistics. *Comput. Oper. Res.* **2012**, *39*, 3215–3228. [CrossRef]
72. Yu, H.; Li, W.; Chen, C.; Liang, J.; Gui, W.; Wang, M.; Chen, H. Dynamic Gaussian bare-bones fruit fly optimizers with abandonment mechanism: Method and analysis. *Eng. Comput. Germany* **2020**, in press. [CrossRef]
73. Yu, H.; Zhao, N.; Wang, P.; Chen, H.; Li, C. Chaos-enhanced synchronized bat optimizer. *Appl. Math. Model.* **2020**, *77*, 1201–1215. [CrossRef]
74. Li, S.; Chen, H.; Wang, M.; Heidari, A.A.; Mirjalili, S. Slime mould algorithm: A new method for stochastic optimization. *Future Gener. Comp. Syst.* **2020**, *111*, 300–323. [CrossRef]
75. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comp. Syst.* **2019**, *97*, 849–872. [CrossRef]
76. Hu, J.; Chen, H.; Heidari, A.A.; Wang, M.; Zhang, X.; Chen, Y.; Pan, Z. Orthogonal learning covariance matrix for defects of grey wolf optimizer: Insights, balance, diversity, and feature selection. *Knowl. Based Syst.* **2021**, *213*, 106684. [CrossRef]
77. Tu, J.; Chen, H.; Liu, J.; Heidari, A.A.; Zhang, X.; Wang, M.; Ruby, R.; Pham, Q.-V. Evolutionary biogeography-based whale optimization methods with communication structure: Towards measuring the balance. *Knowl. Based Syst.* **2021**, *212*, 106642. [CrossRef]
78. Zitzler, E.; Laumanns, M.; Thiele, L. SPEA2: Improving the strength Pareto evolutionary algorithm. In Proceedings of the Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems, Athens, Greece, 19–21 September 2001; pp. 95–100.
79. Cheng, R.; Jin, Y.; Olhofer, M.; Sendhoff, B. Test problems for large-scale multiobjective and many-objective optimization. *IEEE Trans. Cybern.* **2017**, *47*, 4108–4121. [CrossRef] [PubMed]
80. Ma, X.; Liu, F.; Qi, Y.; Wang, X.; Li, L.; Jiao, L.; Yin, M.; Gong, M. A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables. *IEEE Trans. Evol. Comput.* **2015**, *20*, 275–298. [CrossRef]
81. Zhang, X.; Tian, Y.; Cheng, R.; Jin, Y. A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. *IEEE Trans. Evol. Comput.* **2018**, *22*, 97–112. [CrossRef]
82. Liu, R.; Li, J.; Fan, J.; Jiao, L. A dynamic multiple populations particle swarm optimization algorithm based on decomposition and prediction. *Appl. Soft Comput.* **2018**, *73*, 434–459. [CrossRef]
83. Rambabu, R.; Vadakkepat, P.; Tan, K.C.; Jiang, M. A mixture-of-experts prediction framework for evolutionary dynamic multiobjective optimization. *IEEE Trans. Cybern.* **2020**, *50*, 5099–5112. [CrossRef] [PubMed]
84. Guo, Y.; Yang, H.; Chen, M.; Cheng, J.; Gong, D. Ensemble prediction-based dynamic robust multi-objective optimization methods. *Swarm Evol. Comput.* **2019**, *48*, 156–171. [CrossRef]

85. Muruganantham, A.; Tan, K.C.; Vadakkepat, P. Evolutionary dynamic multiobjective optimization via Kalman filter prediction. *IEEE Trans. Cybern.* **2015**, *46*, 2862–2873. [CrossRef] [PubMed]
86. Nebro, A.J.; Ruiz, A.B.; Barba-González, C.; García-Nieto, J.; Luque, M.; Aldana-Montes, J.F. InDM2: Interactive dynamic multi-objective decision making using evolutionary algorithms. *Swarm Evol. Comput.* **2018**, *40*, 184–195. [CrossRef]
87. Gee, S.B.; Tan, K.C.; Alippi, C. Solving multiobjective optimization problems in unknown dynamic environments: An inverse modeling approach. *IEEE Trans. Cybern.* **2016**, *47*, 4223–4234. [CrossRef]
88. Wang, H.; Jiao, L.; Yao, X. Two_Arch2: An improved two-archive algorithm for many-objective optimization. *IEEE Trans. Evol. Comput.* **2014**, *19*, 524–541. [CrossRef]
89. Huband, S.; Barone, L.; While, L.; Hingston, P. *A Scalable Multi-Objective Test Problem Toolkit. Evolutionary Multi-Criterion Optimization*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 280–295.
90. Huband, S.; Hingston, P.; Barone, L.; While, L. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Trans. Evol. Comput.* **2006**, *10*, 477–506. [CrossRef]
91. Das, I.; Dennis, J.E. Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. *SIAM J. Optim.* **1998**, *8*, 631–657. [CrossRef]
92. Zhang, X.; Zheng, X.; Cheng, R.; Qiu, J.; Jin, Y. A competitive mechanism based multi-objective particle swarm optimizer with fast convergence. *Inf. Sci.* **2018**, *427*, 63–76. [CrossRef]
93. Coello, C.A.C.; Lechuga, M.S. MOPSO: A proposal for multiple objective particle swarm optimization. In Proceedings of the 2002 IEEE Congress on Evolutionary Computation, Honolulu, HI, USA, 12–17 May 2002; pp. 1051–1056.
94. D'Angelo, G.; Palmieri, F. GGA: A modified genetic algorithm with gradient-based local search for solving constrained optimization problems. *Inf. Sci.* **2021**, *547*, 136–162. [CrossRef]

mathematics

MDPI

Article

Memetic Strategy of Particle Swarm Optimization for One-Dimensional Magnetotelluric Inversions

Ruiheng Li [1,2], Lei Gao [1,2], Nian Yu [1,2,*], Jianhua Li [3,*], Yang Liu [1,2], Enci Wang [1,2] and Xiao Feng [4]

[1] School of Electrical Engineering, Chongqing University, Chongqing 400044, China; liruiheng@cqu.edu.cn (R.L.); 201911021045@cqu.edu.cn (L.G.); 20116884@cqu.edu.cn (Y.L.); wangenci@cqu.edu.cn (E.W.)
[2] State Key Laboratory of Power Transmission Equipment & System Security and New Technology, Chongqing 400044, China
[3] Key Laboratory of Geophysical Electromagnetic Probing Technologies of Ministry of Natural Resources, Institute of Geophysical and Geochemical Exploration, Chinese Academy of Geological Science, Langfang 065000, China
[4] School of Economics and Business Administration, Chongqing University, Chongqing 400044, China; fengxiao@cqdvan.com
* Correspondence: yunian@cqu.edu.cn (N.Y.); ljianhua@mail.cgs.gov.cn (J.L.)

Abstract: The heuristic algorithm represented by particle swarm optimization (PSO) is an effective tool for addressing serious nonlinearity in one-dimensional magnetotelluric (MT) inversions. PSO has the shortcomings of insufficient population diversity and a lack of coordination between individual cognition and social cognition in the process of optimization. Based on PSO, we propose a new memetic strategy, which firstly selectively enhances the diversity of the population in evolutionary iterations through reverse learning and gene mutation mechanisms. Then, dynamic inertia weights and cognitive attraction coefficients are designed through sine-cosine mapping to balance individual cognition and social cognition in the optimization process and to integrate previous experience into the evolutionary process. This improves convergence and the ability to escape from local extremes in the optimization process. The memetic strategy passes the noise resistance test and an actual MT data test. The results show that the memetic strategy increases the convergence speed in the PSO optimization process, and the inversion accuracy is also greatly improved.

Keywords: particle swarm optimization; magnetotelluric; one-dimensional inversions; geoelectric model; optimization problem

Citation: Li, R.; Gao, L.; Yu, N.; Li, J.; Liu, Y.; Wang, E.; Feng, X. Memetic Strategy of Particle Swarm Optimization for One-Dimensional Magnetotelluric Inversions. *Mathematics* **2021**, *9*, 519. http://doi.org/10.3390/math9050519

Academic Editor: Amir H. Alavi

Received: 23 January 2021
Accepted: 24 February 2021
Published: 2 March 2021

1. Introduction

The magnetotelluric (MT) technology is a geophysical electromagnetic detection method that uses electromagnetic induction signals to detect underground electrical structures [1,2]. The horizontal magnetic field is vertically incident into the Earth, which produces a time-harmonic changing induced electromagnetic field in the ground. When the excitation field source is constant, the electromagnetic field induced in the Earth is determined by the underground electrical structure and frequency [3]. Calculating the induced electromagnetic signal based on the electrical structure and frequency constitutes MT forward modeling, and this process satisfies the Maxwell equations. The process of calculating the geoelectric structure according to the induced electromagnetic signal and frequency is the MT inversion, which is implemented by the optimization method [4].

In the optimization process, the electrical structure is used as the optimization parameter to find the smallest objective function, and the difference between the predicted electromagnetic signal and the observed signal is evaluated by the objective function [5]. When only surface electromagnetic signals can be obtained, the inversion problem is severely underdetermined and has multiple solutions. Model roughness is commonly added as a Lagrangian penalty term to the objective function to address ambiguity [6,7].

However, due to the serious nonlinearity of the MT inversion problem, the commonly used gradient optimization method is slow in the optimization process, and the optimal solution is not accurate. Nonlinear optimization methods based on intelligent algorithms often have better results in solving such nonlinear problems [8,9].

Heuristic algorithms are commonly used to solve such nonlinear problems [10,11]. Several common algorithms, including the simulated annealing method, the Bayesian inversion method and genetic algorithm, have been able to initially solve the MT inversion problem and determine the underground electrical structure through the electromagnetic response signal of the MT method [12,13]. Among these heuristic swarm intelligence algorithms, the particle swarm optimization (PSO) algorithm is widely used in the MT inversion due to its simple implementation and less adjustment parameters [14,15]. With the introduction of the inertia weight factor, the time-varying acceleration factor strategy and the strategy based on reproduction and subgroup hybridization, the shortcomings of PSO—that it easily falls into local extremes and has slow convergence in the later stages of evolution—are gradually improved [16–18]. However, these algorithms still have not overcome the shortcomings of the lack of population diversity and the uncoordination of individual cognition and social cognition capabilities.

With the development of memetic strategies, which take the process of memetic evolution as inspiration, using interactions between intelligent individuals to achieve population evolution and memetic evolution has become an important tool for enhancing population diversity and coordinating individual cognition and social cognition [19,20]. For the MT inversion problem, our strategy is to calculate the cognitive attraction coefficient through sine-cosine mapping to balance individual cognition and social cognition in the optimization process. Then, to further improve convergence in the optimization process and the ability to escape local extremes, we use dynamic inertia weights (DIWs) to integrate the previous experience of the population into the evolutionary process, and we use genetic mutations to enrich the diversity of the population.

Our contributions to the MT inversion with PSO optimization are as follows:

- We use opposition-based learning strategy to search for a suitable initial population of geoelectric model more accurately, the strategy can help to determine the appropriate global optimal search direction in the early stage and accelerate convergence.
- We use DIWs based on sine mapping to integrate empirical cognition of the previous inversion iterations, and this strategy can strengthen the optimization ability of MT inversions.
- We used sine-cosine acceleration coefficients to balance the influence of individual cognition and group cognition on the evolutionary process, this strategy can improve the global optimization capability, and convergence stability in the MT inversion process.

In the remainder of this paper, we first review the background of MT inversions based on PSO in Section 2. Then, we present the proposed memetic strategy in detail in Section 3. This section mainly focuses on the main framework of the memetic strategy, introduces population initialization, uses DIWs to integrate empirical cognition, and uses the cognitive attraction coefficient to accelerate population evolution and population mutation (PM). In Section 4, the inversion effects of the proposed memetic strategy on different geoelectric models are presented. Subsequently, in Section 5, we evaluate the stability of the memetic strategy using a noise immunity test and an actual data test. Finally, conclusions are drawn in Section 6.

2. PSO for 1D MT Inversions

2.1. Forward Modeling

The MT method involves measuring orthogonal components of the electric field **E** and the magnetic field **H** at the Earth's surface (Figure 1a). The electromagnetic field we observe is excited by the natural magnetic field. The frequency is lower than 10^5 Hz so we could ignore the displacement current in the quasi-static approximation of electromagnetic field. When a magnetic field **H** is applied to the ground, it produces an electric field **E** through

electromagnetic induction. The impedance **Z** is used to express the relation between the electromagnetic fields as follows [2]:

$$\begin{bmatrix} \mathbf{E}_x \\ \mathbf{E}_y \end{bmatrix} = \begin{bmatrix} \mathbf{Z}_{xx} & \mathbf{Z}_{xy} \\ \mathbf{Z}_{yx} & \mathbf{Z}_{yy} \end{bmatrix} \begin{bmatrix} \mathbf{H}_x \\ \mathbf{H}_y \end{bmatrix} \tag{1}$$

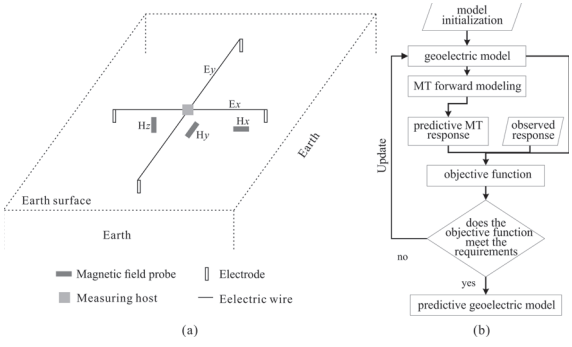


Figure 1. Basic introduction of the magnetotelluric (MT) method. (**a**) shows the layout of the MT signal acquisition system. The electrodes connected by wires is used to obtain electric field data, and the magnetic field probes are used for collecting magnetic field data. The host is used to record the signal at various frequencies. (**b**) The application of the optimization process in MT inversions.

For the one-dimensional case, $\mathbf{Z}_{xx} = 0$, $\mathbf{Z}_{yy} = 0$ and $\mathbf{Z}_{xy} = -\mathbf{Z}_{yx}$. The impedance tensor can be decomposed into two components, corresponding to the apparent resistivity and the phase. For an N-stratum geoelectric model, the apparent resistivity ρ_ω and the phase φ can be derived from the impedance **Z** regardless of the orientations of the x and y axes as follows:

$$\begin{gathered} \rho_\omega = \frac{|Z_1|^2}{\omega\mu} \quad \varphi = \tan^{-1}\frac{\mathrm{Im}(Z_1)}{\mathrm{Re}(Z_1)} \\ Z_m = Z_{om}\frac{1-L_{m+1}e^{-2k_m h_m}}{1+L_{m+1}e^{-2k_m h_m}} \quad L_{m+1} = \frac{Z_{om}+Z_{m+1}}{Z_{om}+Z_{m+1}} \\ Z_N = Z_{oN} \quad Z_{om} = -i\omega\mu/k_m \quad k_m = \sqrt{-i\mu\sigma_m\omega} \end{gathered} \tag{2}$$

where Z_m is the impedance at the top of the mth stratum, Z_{om} is the intrinsic impedance of the mth stratum, the magnetic permeability μ is assigned its free space value and ω is the angular frequency. For the mth stratum, ρ_m is the resistivity and h_m is the thickness. Usually, the apparent resistivity is the observed response that is used to obtain the geoelectric model through inversion.

2.2. Inversions

The MT inversion problem is an optimization problem in which the objective is to predict a model that is close to the real geoelectric structure from the observed response (Figure 1b). The optimization process update the geoelectric model iteratively to find the minimum objective function. The objective function of this optimization problem can be divided into two terms, one corresponding to data fitting and one corresponding to the model smoothness [18]. The data fitting term measures the difference between the observed response and the predicted response (Figure 1b). The smoothness term measures the change in the magnitude of the resistivity of each stratum [21]. The objective function can be expressed as follows:

$$\min \Phi(\mathbf{m}) = \min\left(\lambda\|\mathbf{C}_m\mathbf{m}\|^2 + \|\mathbf{C}_d(\mathbf{F}[\mathbf{m}] - \mathbf{d})\|^2\right) \tag{3}$$

where $\Phi(\mathbf{m})$ is the objective function; $\mathbf{F}$ is the forward modeling operator; $\mathbf{d}$ represents the observed data; $\mathbf{C}_m$ and $\mathbf{C}_d$ are the covariance matrices of the model vector and the observed data vector, respectively; and λ is the Lagrange multiplier weighting the model smoothness term relative to the total norm. The objective function is updated with the predicted model $\mathbf{m}$, and its value gradually decreases in each iteration.

To minimize the objective function, several iterative methods of linear inversions have been proposed [22,23]. The occam's inversion is a popular and stable inversion algorithm based on an iterative method in which the model is directly updated in each iteration, causing the value of the objective function to decrease steadily [24–26]. The model is updated as follows:

$$\begin{gathered}\mathbf{m}_{k+1} = \left[\frac{1}{\lambda}\mathbf{C}_m^T\mathbf{C}_m + (\mathbf{C}_d\mathbf{J}_k)^T\mathbf{C}_d\mathbf{J}_k\right]^{-1}(\mathbf{C}_d\mathbf{J}_k)^T\mathbf{C}_d\mathbf{d}_g \\ \mathbf{d}_g = \mathbf{d} - \mathbf{F}[\mathbf{m}_k] + \mathbf{J}_k\mathbf{m}_k\end{gathered} \tag{4}$$

The iteration process begins with an initial model guess $\mathbf{m}_0$, and the model is updated to $\mathbf{m}_k$ in the kth iteration. The optimal model is considered to be found when a maximum number of iterations, a convergence threshold for the objective function or some other termination criterion is reached. In addition, it is important to note that the model parameters are typically expressed in terms of the logarithm of the resistivity in order to reduce the variations in the gradient.

The linear inversion methods can easily become trapped in local minima and require considerable computational effort to calculate the gradient of the objective function [24]. Moreover, they are critically dependent on the initial model [27]. However, global optimization methods based on heuristic algorithms overcome these shortcomings [8,28].

2.3. PSO Optimization

The PSO algorithm will make the population evolve more intelligently after each iteration and can accumulate search knowledge, which is called an evolutionary algorithm [29–31]. The PSO algorithm does not use the survival of the fittest but uses a mechanism in which each individual in the population competes with the others to generate the global optimal solution. It generates the optimal solution through information sharing and a mechanism of cooperation between the individuals in the population [32,33].

Suppose the PSO consists of multiple particles. In the D-dimensional search space, the particle swarm contains n particles. The position of the nth particle in the D-dimensional space is defined as x_i:

$$x_i = (x_{i1}, x_{i2}, \cdots, x_{iD}), \quad i = 1, 2, \cdots, n. \tag{5}$$

Suppose the current velocity of particle x_i and its individual optimal historical position are v_i and p_i, respectively, as follows:

$$\begin{gathered}v_i = (v_{i1}, v_{i2}, \cdots, v_{iD}) \\ p_i = (p_{i1}, p_{i2}, \cdots, p_{iD}).\end{gathered} \tag{6}$$

Then, for the entire particle swarm, the global optimal position is P_g:

$$p_g = (p_{g1}, p_{g2}, \cdots, p_{gD}). \tag{7}$$

At the tth moment, the velocity update formula of the dth dimension of particle x_i is:

$$v_{id}(t+1) = \omega v_{id}(t) + c_1 r_1 (p_{id}(t) - x_{id}(t)) + c_2 r_2 (p_{gi}(t) - x_{id}(t)), \tag{8}$$

where ω is the inertia weight and is in the range [0, 1]. c_1 and c_2 are the acceleration coefficients. r_1 and r_2 are random coefficients, both of which are in [0, 1], which determine the motion of semirandom particles affected by the single and global optimal solutions.

The particle velocity update process has three main parts: the current initial velocity, the self-motion trajectory and self-trajectory correction. The influence of the current speed on the particle update speed can be adjusted by the inertia weight. The influence of the particle's own trajectory on the particle update speed can be adjusted by the acceleration coefficients and the random coefficients. When the trajectory is inaccurate, it needs to be corrected with the help of global optimization.

For particle x_i, we can update the position x_{id} of the dth dimension according to the velocity:

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1). \tag{9}$$

3. Memetic Strategies

If the particle swarm is regarded as a social population, the three parts of the particle update speed reflect the balance of the population with respect to the global optimum and the local optimum. The particle update speed can be regarded as the cognition of the social population in the evolutionary direction. The evolutionary update speed at the current moment t can provide a reference basis for the evolutionary update speed at the next moment $t + 1$. During evolution, the evolution of a single particle needs to refer to its own previous evolutionary state and the evolutionary state of the population. The main advantage of the PSO algorithm is that it is simple, effective and easy to implement. However, the PSO algorithm faces premature convergence and easily falls into a local optimal solution [34].

Compared with other group-based methods, the ability of PSO to micromediate and avoid local optima is weaker, which is mainly due to the lack of diversity in the search process [35]. Therefore, we improve the evolutionary diversity of the population in the search process; the basic flow of the cultural strategy is shown in Figure 2.

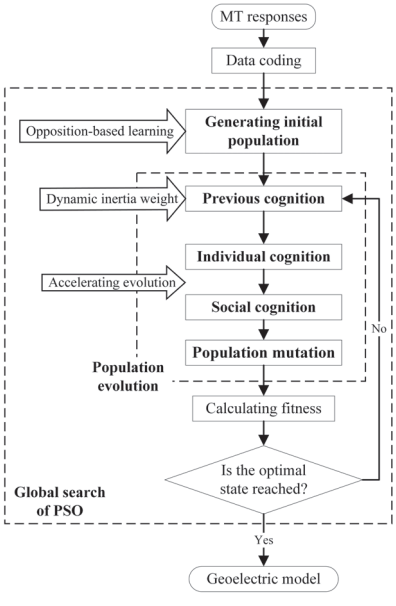


Figure 2. Flow chart of memetic strategy for MT inversions.

3.1. Framework

In the memetic strategy, the overall optimization search uses PSO. To optimize an objective function, the first task is to generate the initial population. The initial population in PSO is randomly generated, which may affect the convergence speed of the algorithm

and the accuracy of the final solution. In the absence of prior knowledge, we use a method based on opposition learning to replace the random initial population positions as a new initial population strategy. This can increase the chance of reaching the global optimal solution [36].

In the entire iterative search process, we hope that the search range in the early stage is as large as possible to enhance the global optimization capability. We hope that the search range in the search period does not change greatly in order to enhance the local optimization capability. These factors mean that we need to change the inertia weight to adjust the first part of the right-hand term of Equation (8). The dynamic inertia weight is applied to the previous population cognition to provide a reference for the optimization process in the current iteration.

In population evolution, the second part of the right term in Equation (8) represents the synchronization between the current position of the particle and the optimal position of the individual in the iteration. This is the process of the particle revising its own evolutionary path, reflecting the effect of the particle's own evolutionary experience on its own next evolution. The third part of the right-hand term of Equation (8) shows the process of synchronizing the current position of the particle with the best position of the group. This is the corrective behavior of the particle after observing the evolution of the surrounding particles. This kind of social behavior reflects the group's information sharing and cooperation.

After obtaining the best population in the current iteration, to further enhance the diversity of the population, we set the mutation of the individual particles for the population. The aim of this operation is to make the optimization process converge stably while maintaining a certain ability to jump out of a local optimum.

3.2. Population Initialization

When there is no prior information, the initial population is usually randomly generated, which often leads to revisiting a hopeless area in the search space [37]. Opposition-based learning (OBL) considers candidate solutions as well as their opposite solutions [38]. OBL introduces a random solution and its corresponding inverse solution, which can yield more than two independent solutions. This randomly generates more promising solutions. OBL has been successfully applied in various population-based evolutionary algorithms [39,40]. To effectively increase the diversity of the initial population, we use the OBL strategy to generate the initial population, which includes two types of populations: a random initial population and an anti-population. The random initial population $\{x_{id}, d = 1, 2, \ldots, D\}$ is generated randomly according to the form shown in Equation (5). Supposing the inverse population is $\{x'_{id}, d = 1, 2, \ldots, D\}$, x'_{id} can be expressed as:

$$x'_{id} = x_{max,d} + x_{min,d} - x_{id}, \tag{10}$$

where $x_{max,d}$ and $x_{min,d}$ are the maximum and minimum values of the dth dimension of particle x_i in the D-dimensional search space. After merging the random initial population and the antipopulation, we select the particles with less fitness to form a new initial population.

3.3. Dynamic Inertia Weight

The inertia weight ω can limit the search range of particles, which allows the particles to maintain inertia of motion and search in a new area. This means that this new evolution includes old evolutionary habits and experiences. When the inertial weight is relatively high, the new evolution can eliminate the influence of the previous evolutionary experience. This is conducive to expanding the search field, but the convergence speed can easily slow in optimization. When the inertia weight is relatively small, the particle maintains its evolutionary direction based on previous experience. When the evolutionary direction is correct, this will help speed up the convergence rate of the global optimization, but it is easy for the optimization to fall into local extremes [35].

The inertial weight ω affects the search speed and accuracy. For optimization problems with severe nonlinearity, the use of fixed inertia weights will result in fast convergence, and global optimization is often impossible. For the algorithm to obtain the best results in the optimization process, varying inertia weights need to be used. Using a linearly decreasing inertia weight is a traditional variable inertia weight strategy that can optimize performance well [16]. When the initial inertia weight value is large, the optimal solution range can be found quickly; then, the inertia weight value decreases, and the particles begin to search more finely.

However, because the slope is constant, the speed change always remains at the same level. If the initial iteration does not produce better points, then the accumulation of iterations and the rapid decay of speed may lead to a final local optimal value. Therefore, we use a nonlinear strategy, sine mapping, with ergodicity, nonrepetition and irregularity [34,41] to adjust the inertia weight ω of PSO. This strategy can not only enhance the population diversity in the search process but also enhance the ability to converge to the global optimum. The dynamic inertia weight based on sine mapping can be expressed as:

$$\omega = k_t = \frac{q}{4}\sin(\pi k_t - 1), \quad k_t \in (0,1), \quad t = 1,2,3,\ldots,T, \tag{11}$$

where the range of q is from 0 to 4.

3.4. Accelerating Evolution

The acceleration coefficients c_1 and c_2 (Equation (8)) are the cognitive attraction coefficients in the optimization process, and the optimization of the group is controlled by the learning situation. In the early stage of the optimization process, the particles need strong self-cognition and weak social cognition. The global search function is more important. At this time, the particle can traverse as many local extremes as possible in the search space.

In the later stage of optimization, the particles must have strong social cognition and weak self-awareness to avoid falling into local extremes in optimization. The values of the cognitive attraction coefficients reflect the degree of influence of the information exchange on particles, and the information exchange includes the experiential information of the particle itself and the global optimal information. Setting the learning factor to a value that is too large or too small is not conducive to the optimization of the particles, so it is necessary to balance the evolution speed of the particles in the early and late stages of the optimization process.

The enhancing effect of sine-cosine mapping on population diversity and convergence in the optimization process can be used to improve this linear asynchronous strategy. We use sine-cosine acceleration coefficients (SCACs) to adjust the balance between individual cognition and social cognition [42]. The cognitive attraction coefficient can be expressed as:

$$\begin{aligned} c1 &= \alpha \times \sin((1 - \tfrac{t}{T}) \times \tfrac{\pi}{2}) + \delta \\ c2 &= \alpha \times \cos((1 - \tfrac{t}{T}) \times \tfrac{\pi}{2}) + \delta, \end{aligned} \tag{12}$$

where the constants α and δ are 2 and 0.5, respectively.

3.5. Population Mutation

To enhance the ability to jump out of local extremes, we introduce the mutation operator from the genetic algorithm into the PSO [43]. This can expand the search space of the particles themselves, enhance the diversity of the population and further increase the possibility of finding the optimal solution. The particles will reset with a certain probability after each evolution. When the mutation condition is met, the mutation jumps out of the current position; otherwise, the original position remains unchanged. The particle variation can be expressed as:

$$x_{id}(t) = r \times (\mathbf{U}_x - L_x)/n + (\mathbf{U}_x - L_x)/2, \tag{13}$$

where r is uniformly distributed in the range $[-1, 1]$, $\mathbf{U}_x$ and L_x are the upper and lower limits of a given position and n is 4. The mutation condition is random mutation, and the mutation probability is 10%.

3.6. Fitness

The fitness in the optimization process refers to the objective function setting in MT inversions. The L2 norm is used to define the misfit between the observed MT response data and the predicted response data. The fitness can be expressed as:

$$\begin{aligned} fit &= c_{rho} fit_{rho} + c_{phi} fit_{phi} \\ &= c_{rho} \left\| 1 - \rho_{pred} \Big/ \rho_{obs} \right\|^2 + c_{phi} \left\| 1 - \varphi_{pred} \Big/ \varphi_{obs} \right\|^2 \end{aligned} \tag{14}$$

where the overall fitness is composed of apparent resistivity fitness and phase fitness model fitness. Their weight coefficients are c_{rho} and c_{phi}. Since the apparent resistivity and phase are variables with different units, in order to transform the apparent resistivity and phase fitness into a unified dimension, we normalize the response data and prediction data.

4. Test Model

We designed two common geoelectric models, a three-layer model and a five-layer model. These models were used to generate synthetic MT response data. Different PSO methods predicted the geoelectric models based on these response data, and the MT responses were obtained through MT forward modeling. Comparing the geoelectric model and the responses predicted by different methods allowed us to test the effect of our memetic strategy.

4.1. Three-Layer Model

The three-layer geoelectric model and its MT response are shown in Figure 3. The resistivity values of the geoelectric model are 100 Ω·m, 20 Ω·m and 100 Ω·m. The thicknesses of the geoelectric model are 100 m, 200 m and infinity. We used this model to generate an MT response including the apparent resistivity and phase, which was the supposed response. The geoelectric model predicted by the optimization method based on the supposed MT response contains five values, namely, the resistivity values of the three layers in the geoelectric model and the thickness values of the first two layers. The predicted geoelectric model can be used to regenerate the apparent resistivity and phase response through MT forward modeling.

In the optimization process, the population size is 100. For the supposed three-layer geoelectric model, both traditional PSO and our strategy can obtain good results, but our strategy predicts the results more accurately. From the comparison of the geoelectric models (Figure 3a), the resistivity value of the second layer of the geoelectric model predicted by traditional PSO is less than the supposed value, and the depth value at the bottom of this layer is slightly larger. The resistivity values of the first layer and the third layer predicted by traditional PSO are smaller than the supposed value, and the misfit of the two strata is not as large as the misfit of the second stratum.

From the comparison of the MT responses (Figure 3b,c), the misfit between the MT responses predicted by traditional PSO and the supposed responses is larger. The large misfit is mainly concentrated in the low- and high-frequency ranges. This misfit is more obvious on the apparent resistivity curve. The responses predicted by our strategy, the apparent resistivity curve and the phase curve, perfectly match the supposed response curve.

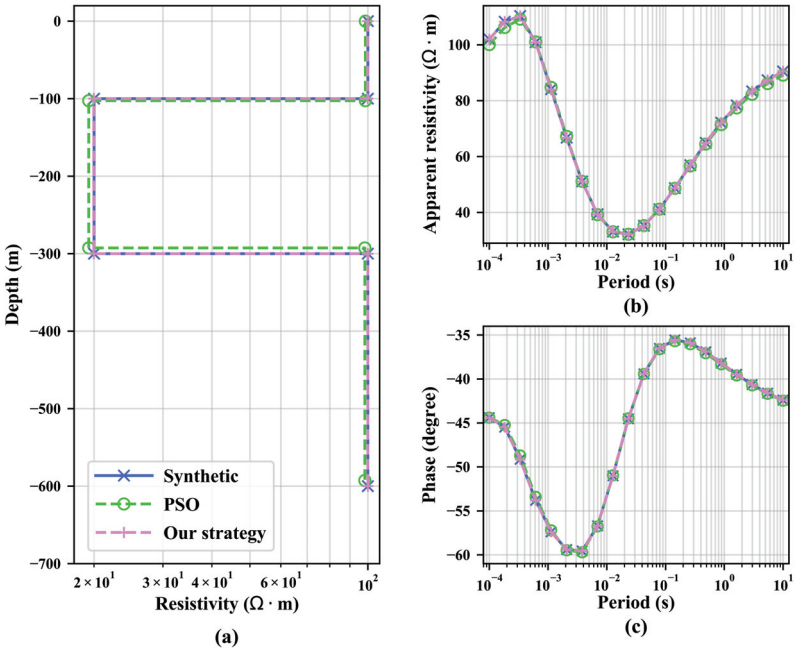


Figure 3. The three-layer geoelectric model and its MT response predicted by traditional PSO and by our strategy. (**a**–**c**) represent the geoelectric model, apparent resistivity responses and represent phase responses, respectively. The blue lines represent the supposed three-layer geoelectric model and its MT responses. The green lines represent the geoelectric model predicted by the traditional PSO and its MT responses. The purple lines represent the geoelectric model predicted by our strategy and its MT responses.

A detailed comparison of the low-frequency and high-frequency parts is shown in Figure 4. For the apparent resistivity curve, the responses predicted by traditional PSO show a large misfit starting at 10^4 Hz. In the 10^4 Hz–$10^{2.5}$ Hz interval, this deviation is very obvious (Figure 4a). In the range of 10^0 Hz–10^{-1} Hz, the misfit of the response predicted by traditional PSO decreases, and it gradually increases as the frequency decreases (Figure 4b). For the same low-frequency and high-frequency ranges, the characteristics of the misfit of apparent resistivity are different. In the low-frequency range, the misfit of traditional PSO always exists and is not concentrated in the 10^4 Hz–$10^{2.5}$ Hz range, similar to the misfit of the apparent resistivity curve (Figure 4c). The deviation of the response predicted by traditional PSO has the same characteristics in the high-frequency range (Figure 4d).

For the traditional PSO method, our memetic strategy has four improved steps, namely, group initialization with OBL, using DIWs to integrate empirical cognition, using the cognitive attraction coefficient to accelerate population evolution and PM. We call them PSO-OBL, PSO-OBL-DIW, PSO-OBL-DIW-SCAC and PSO-OBL-DIW-SCAC-PM. Combining traditional PSO and these four improvements, the corresponding optimization process is shown in Figure 5.

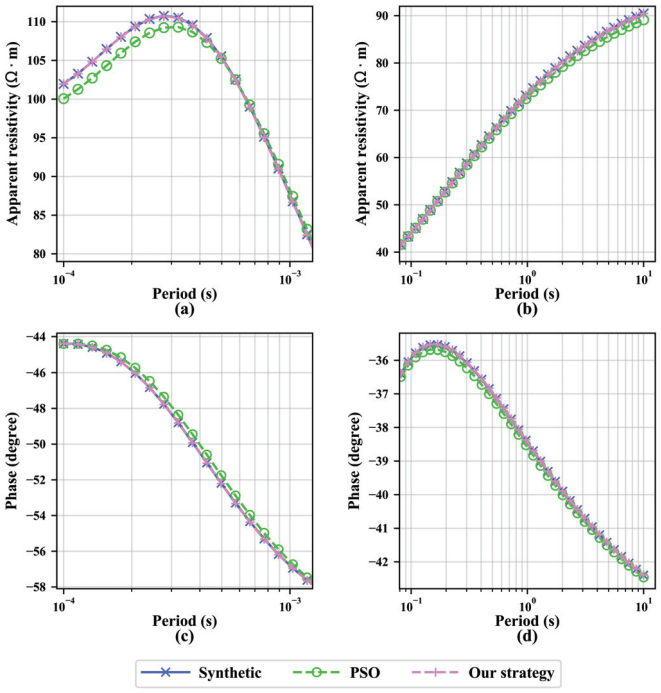


Figure 4. Comparison of the MT response in special frequency bands for the three-layer model. (**a**,**b**) represent apparent resistivity curves, and (**c**,**d**) represent phase curves. The blue lines represent the supposed MT responses. The green lines represent the MT responses predicted by traditional PSO. The purple lines represent the MT responses predicted by our strategy.

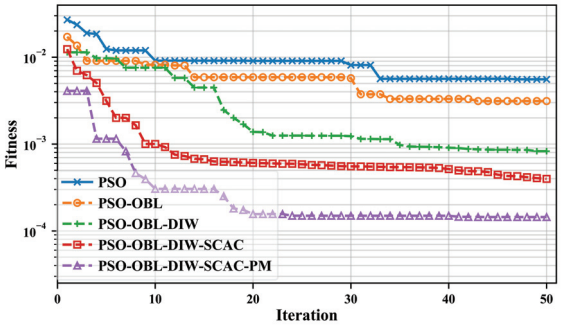


Figure 5. Comparison of the optimization process of different strategies in the three-layer geoelectric model test. The number of evolutionary iterations is 50. The blue line represents the optimization process of traditional PSO. The orange line represents the optimization process of PSO-opposition-based learning (OBL). The green line represents the optimization process of PSO-OBL-dynamic inertia weight (DIW). The red line represents the optimization process of PSO-OBL-DIW-sine-cosine acceleration coefficient (SCAC). The violet line represents the optimization process of PSO-OBL-DIW-SCAC-population mutation (PM).

Using OBL can determine the appropriate initial population more accurately, which can enable the search process to find the appropriate global optimal search direction in the early stage and accelerate convergence. At the early stage, the fitness decline rate

of PSO-OBL is faster than that of traditional PSO. Adding DIWs based on sine mapping can enable the evolution of the population to better combine with previous cognitive experience. Therefore, after 17 iterations, the fitness decline rate of PSO-OBL-DIW is faster than that of PSO-OBL.

On the basis of PSO-OBL-DIW, the advantages of SCACs in effectively integrating individual experience and group experience are used to reflect the faster fitness decline rate of PSO-OBL-DIW-SCAC. The final fitness also remained at a low level. After the population evolution, the population was allowed to continue to produce genetic mutations, which can further accelerate the convergence of the optimization process. The final fitness of PSO-OBL-DIW-SCAC-PM was generally lower than that of PSO-OBL-DIW-SCAC.

The accuracy comparison of the three-layer geoelectric models predicted by different methods is shown in Table 1. The misfit between the predicted value and the supporting value can be expressed as the absolute value of the normalized error. The misfit trends of different methods are consistent with the final fitness trend of optimization. Each improvement increases the prediction accuracy of the resistivity value and the thickness value in the geoelectric model.

Table 1. Accuracy comparison of three-layer geoelectric model predicted by different methods.

		ρ(Ω·m)			h(m)		Fitness
		ρ_1	ρ_2	ρ_3	h_1(m)	h_2(m)	
Supposed model		100.00	20.00	100.00	100.00	200.00	
PSO	model	98.61	19.38	98.27	102.58	189.95	3.32×10^{-3}
	misfit [1]	1.39	3.08	1.73	2.58	5.02	
PSO-OBL	model	99.84	20.42	100.12	97.36	205.08	1.95×10^{-3}
	misfit	0.16	2.09	0.12	2.64	2.54	
PSO-OBL-DIW	model	100.16	20.37	100.21	98.21	205.27	5.94×10^{-4}
	misfit	0.16	1.87	0.21	1.78	2.64	
PSO-OBL-DIW-SCAC	model	100.05	20.12	100.06	99.41	201.61	4.03×10^{-4}
	misfit	0.05	0.58	0.06	0.59	0.81	
PSO-OBL-DIW-SCAC-PM	model	99.96	19.97	100.03	200.15	99.86	2.39×10^{-4}
	misfit	0.04	0.17	0.03	0.08	0.14	

[1] The misfit = $|v_{pred} - v_{supp}|/v_{supp}$, v_{pred} is the predictive value, v_{supp} is the parameter of the supposed model.

4.2. Five-Layer Model

Our method is suitable not only for three-layer models but also for more complex five-layer models. The five-layer geoelectric model and its MT responses are shown in Figure 6. The resistivity values of the geoelectric model are 100 Ω·m, 20 Ω·m, 200 Ω·m, 50 Ω·m and 100 Ω·m. The thicknesses of the geoelectric model are 1000 m, 500 m, 1000 m, 2000 m and infinity. The geoelectric model predicted by the optimization method based on the supposed MT responses contains nine values, namely, the resistivity values of the five layers in the geoelectric model and the thickness values of the first four layers.

For the supposed five-layer geoelectric model, both traditional PSO and our strategy can achieve good results, but our strategy predicts the results more accurately. From the comparison of geoelectric models (Figure 6a), the resistivity value of the third layer of the geoelectric model predicted by traditional PSO is greater than the supposed value, and the top depth and the bottom depth are both larger. The bottom depth of the fourth layer of the geoelectric model predicted by traditional PSO is obviously smaller than the supposed value, and the misfit reaches approximately 200 m.

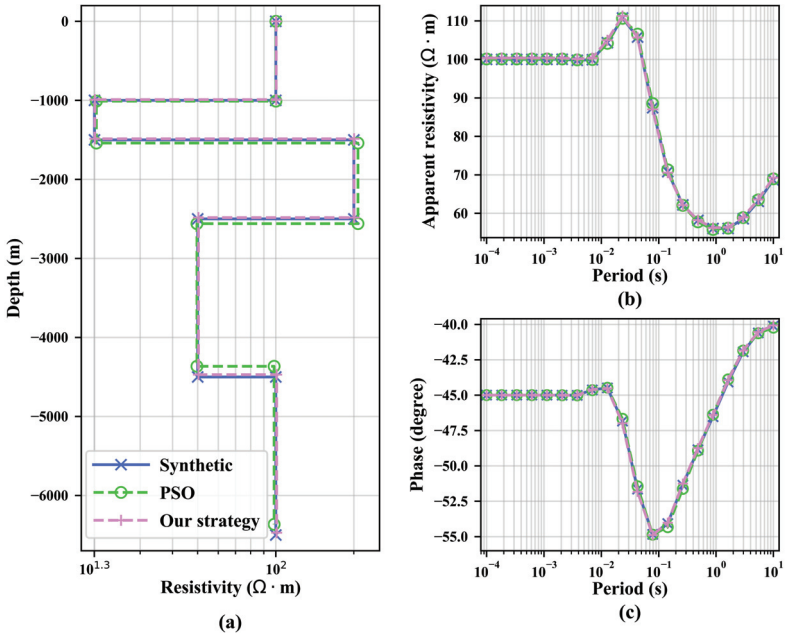


Figure 6. The five-layer geoelectric model and its MT response predicted by traditional PSO and our strategy. (**a**–**c**) represent the geoelectric model, apparent resistivity responses and represent phase responses, respectively. The blue lines represent the supposed three-layer geoelectric model and its MT responses. The green lines represent the geoelectric model predicted by traditional PSO and its MT responses. The purple lines represent the geoelectric model predicted by our strategy and its MT responses.

From the comparison of the MT responses (Figure 6b,c), the responses predicted by traditional PSO have a greater misfit with the supposed responses, and this deviation is mainly concentrated in the mid-frequency range. Similar to the results of the three-layer model (Figure 4b,c), the responses predicted by our strategy, the apparent resistivity curve and the phase curve, perfectly match the assumed response curve.

A detailed comparison of the middle frequency range is shown in Figure 7. The middle frequency interval can be divided into two subintervals for evaluation: the 50 Hz–1 Hz interval (Figure 7a,c) and the $10^{0.8}$ Hz–$10^{-0.3}$ Hz interval (Figure 7b,d). In the first interval, the apparent resistivity and phase misfit of traditional PSO are not concentrated in a certain frequency range but have wide coverage. In the second interval, the apparent resistivity misfit of traditional PSO is mainly concentrated in the middle frequency range, and the phase misfit is mainly concentrated in the higher frequency range.

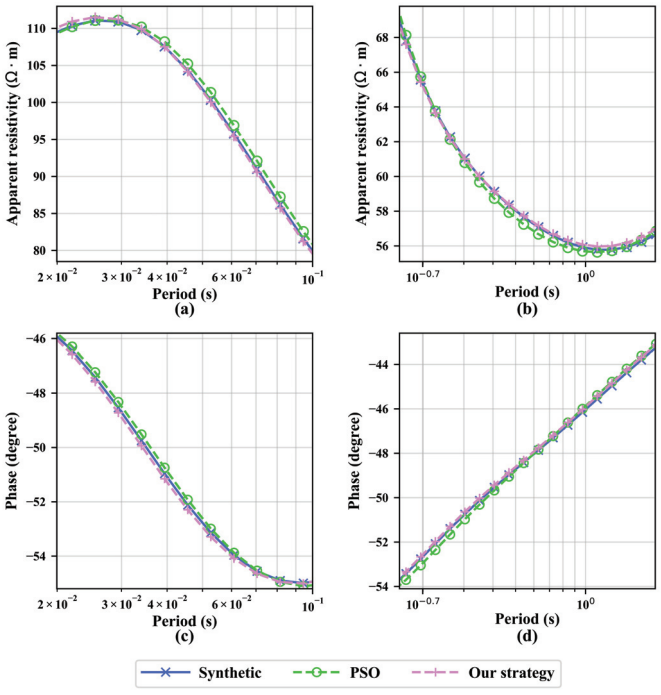


Figure 7. Comparison of the MT response in special frequency bands for the five-layer model. (**a**,**b**) represent apparent resistivity curves, and (**c**,**d**) represent phase curves. The blue lines represent the supposed MT responses. The green lines represent the MT responses predicted by traditional PSO. The purple lines represent the MT responses predicted by our strategy.

For the supposed MT responses of the five-layer geoelectric model, we use traditional PSO, PSO-OBL, PSO-OBL-DIW, PSO-OBL-DIW-SCAC and PSO-OBL-DIW-SCAC-PM to optimize the misfit of the supposed responses and predicted responses. The corresponding optimization process is shown in Figure 8.

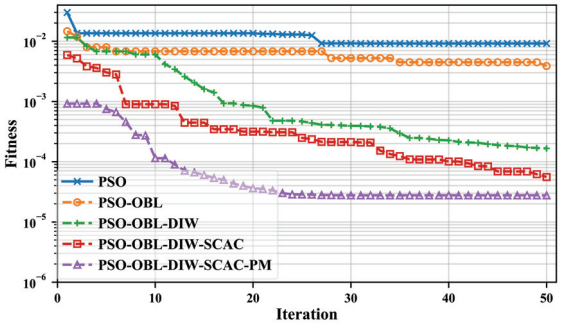


Figure 8. Comparison of the optimization process of different strategies in the five-layer geoelectric model test. The number of evolutionary iterations is 50. The blue line represents the supposed MT response. The green line represents the optimization process of traditional PSO. The yellow line represents the optimization process of PSO-OBL. The green line represents the optimization process of PSO-OBL-DIW. The red line represents the optimization process of PSO-OBL-DIW-SCAC. The red line represents the optimization process of PSO-OBL-DIW-SCAC-PM.

OBL allows the optimization process to find the appropriate global optimal search direction at an early stage and speed up the convergence. This shows the effectiveness of using OBL to determine a suitable initial population. The advantages of PSO-OBL-DIW began to manifest after the 10th evolutionary iteration, indicating that the combination of PSO-OBL-DIW with previous evolutionary cognitive experience is conducive to obtaining a more accurate evolutionary direction for the population. The effective integration of individual experience and group experience through SCACs is still obvious in promoting the optimization process. PSO-OBL-DIW-SCAC changed after the fifth evolutionary iteration to accelerate the convergence speed of fitness and keep the convergence trend stable. Although we set only a 10% gene mutation probability, PSO-OBL-DIW-SCAC-PM still has a great advantage in fitness convergence over PSO-OBL-DIW-SCAC.

Table 2 shows an accuracy comparison of different methods used to predict the five-layer geoelectric model. The misfit between the predicted value and the supposed value can be expressed as the absolute value of the normalized error. The number of parameters of the five-layer geoelectric model is greater than that of the three-layer geoelectric model, and the accuracy of the resistivity and thickness values predicted in the five-layer geoelectric model test is not as good as that in the three-layer geoelectric model test. However, the effect of each improvement on the prediction accuracy is consistent with the effect in the three-layer geoelectric model test. This shows that the effect of our memetic strategy can be applied to a more complex five-layer geoelectric model.

Table 2. Accuracy comparison of five-layer geoelectric model predicted by different methods.

		$\rho(\Omega\cdot\mathbf{m})$					$h(\mathbf{m})$				**Fitness**
		ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	$h_1(\mathbf{m})$	$h_2(\mathbf{m})$	$h_3(\mathbf{m})$	$h_4(\mathbf{m})$	
Supposed model		100.00	20.00	200.00	50.00	100.00	1000.00	500.00	1000.00	2000.00	
PSO	model	100.11	20.34	207.38	49.67	98.51	1009.82	530	1019.5	1806.08	8.60×10^{-3}
	misfit [1]	0.11	1.68	3.69	0.67	1.49	0.98	6	1.95	9.7	
PSO-OBL	model	99.79	20.21	200.52	48.3	100.81	973.19	483.79	968.88	1984.67	5.10×10^{-3}
	misfit	0.21	1.04	0.26	3.39	0.81	2.68	3.24	3.11	0.77	
PSO-OBL-DIW	model	99.53	19.96	195	50.4	98.03	1012.72	510.31	1008.9	1977.88	1.72×10^{-4}
	misfit	0.47	0.22	2.5	0.79	1.97	1.27	2.06	0.89	1.11	
PSO-OBL-DIW-SCAC	model	100	20.31	198.23	50.32	100.04	996.66	508.88	990.29	2021.54	9.18×10^{-5}
	misfit	0	1.53	0.88	0.64	0.04	0.33	1.78	0.97	1.08	
PSO-OBL-DIW-SCAC-PM	model	100.36	20.06	200.48	50.34	100.8	990.26	496.39	996.2	1988.36	2.52×10^{-6}
	misfit	0.36	0.3	0.24	0.69	0.8	0.97	0.72	0.38	0.58	

[1] The misfit = $|v_{pred} - v_{supp}|/v_{supp}$, v_{pred} is the predictive value, v_{supp} is the parameter of the supposed model.

5. Stability Evaluation

To evaluate the stability of our strategy for MT inversions, we conducted a noise immunity test and a test with actual data. In the tests, we compared our improvement strategy with traditional PSO.

5.1. Noise Immunity Test

In the noise immunity test, we designed three different levels of random noise and added noise to the supposed MT responses of the three-layer geoelectric model and the five-layer geoelectric model. The noise levels are 5%, 10% and 15%, respectively. The MT responses of the three-layer geoelectric model with noise are shown in Figure 9.

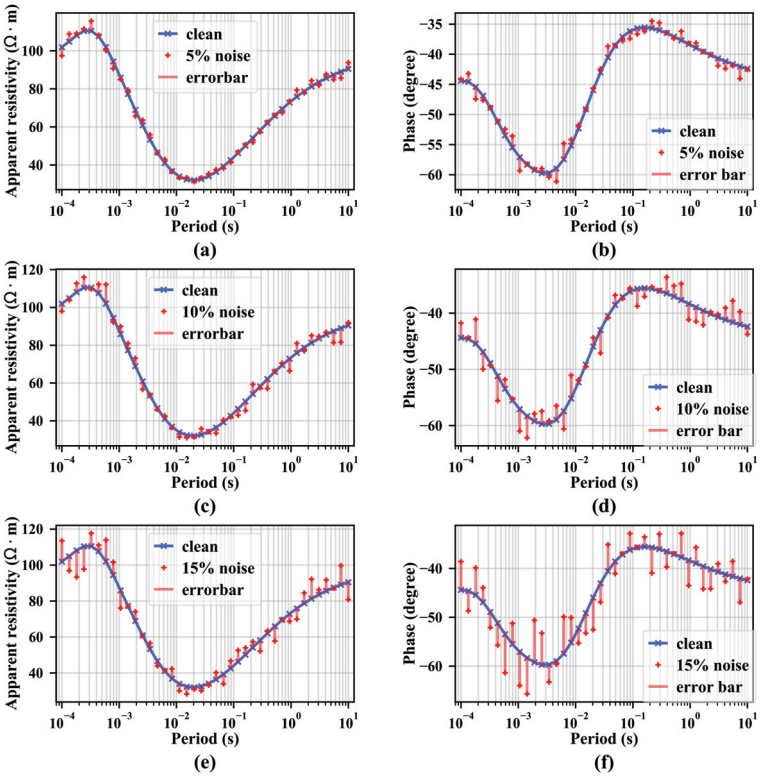


Figure 9. Supposed MT responses of a three-layer geoelectric model with different levels of noise. (**a**,**b**) represent the MT responses when the noise level is 5%. (**c**,**d**) represent the MT responses when the noise level is 10%. (**e**,**f**) represent the MT responses when the noise level is 15%. (**a**,**c**,**e**) represent the apparent resistivity responses. (**b**,**d**,**f**) represent the apparent resistivity responses. The blue lines represent the clean supposed responses. The red dots indicate the noisy responses. The red lines are the error bars between the noisy data and clean data.

For the responses of the three-layer geoelectric model, 5% of the noise has basically no effect on our final prediction results. The predicted geoelectric model and responses perfectly match the assumed geoelectric model and responses. From the comparison of the geoelectric models (Figure 10a), 10% noise and 15% noise cause the predicted resistivity value of the first layer of the geoelectric model to deviate, and the greater the noise is, the greater the deviation. However, the thickness of the first layer is basically consistent with the supposed value. The predicted resistivity value of the second layer is also not affected by noise, but 15% noise makes the predicted thickness smaller than the supposed value. The predicted resistivity value in the third layer corresponding to 10% noise and 15% noise is smaller than the supposed value. Table 3 shows the accuracy comparison of the detailed predictive electrical model.

From the comparison of the MT responses (Figure 10b,c), the predicted value misfit caused by 10% noise and 15% noise is in the same frequency range. For the apparent resistivity curves, the misfit is concentrated in the low- and high-frequency regions, and the misfit in the low-frequency region is larger than that in other regions (Figure 10b). For the phase curves, the misfit is concentrated in the low-frequency to mid-frequency region, and the misfit in the high-frequency data is milder (Figure 10c).

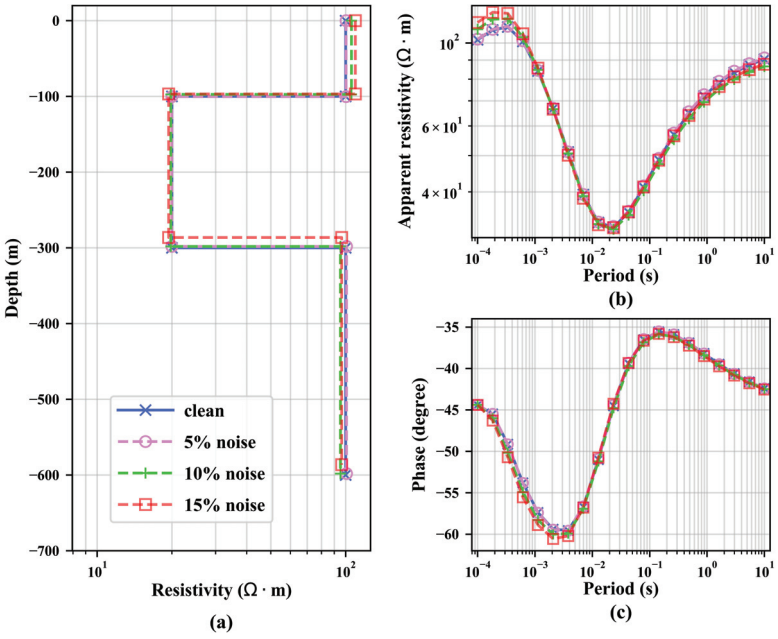


Figure 10. The predicted three-layer geoelectric models and their MT responses under noisy conditions. (**a**–**c**) represent the geoelectric model, apparent resistivity responses and phase responses, respectively. The blue lines represent the supposed three-layer electrical model and clean MT responses, and the purple lines represent the predicted geoelectric model and its MT responses when the noise level is 5%. The green lines represent the predicted geoelectric model and its MT responses when the noise level is 10%. The red lines represent the predicted geoelectric model and its MT responses when the noise level is 15%.

Table 3. Comparison of the accuracy of the predicted three-layer geoelectric model under different noise levels.

		$\rho(\Omega\cdot\mathbf{m})$			$h(\mathbf{m})$		Fitness
		ρ_1	ρ_2	ρ_3	$h_1(\mathbf{m})$	$h_2(\mathbf{m})$	
Supposed model		100.00	20.00	100.00	200.00	100.00	
5% noise	model	100.42	19.98	100.8	100.25	197.8	2.19×10^{-2}
	misfit [1]	0.42	0.09	0.8	0.24	1.1	
10% noise	model	105.6	19.8	95.34	97.31	200.7	4.25×10^{-2}
	misfit	5.6	0.98	4.66	2.69	0.36	
15% noise	model	109.44	19.4	96.43	96.72	189.6	6.63×10^{-2}
	misfit	9.44	2.99	3.57	3.28	5.2	

[1] The misfit = $|v_{pred} - v_{supp}|/v_{supp}$, v_{pred} is the predictive value, v_{supp} is the parameter of the supposed model.

For the MT response of the synthetic five-layer geoelectric model, the response after adding noise is shown in Figure 11. For the noise-containing responses of the five-layer geoelectric model, the influence of noise is obviously greater than that in the three-layer model. From the comparison of the geoelectric models (Figure 12a), the predicted resistivity values are close to the supposed values in the first three layers. For the second stratum, the predicted value for 5% noise has a slight deviation. 10% noise and 15% noise increase the deviation. The maximum misfit of the predicted resistivity of 15% noise is close to 15%, and the maximum misfit of the predicted layer thickness is close to 10%. Table 4 shows the accuracy comparison of the detailed predictive geoelectric model.

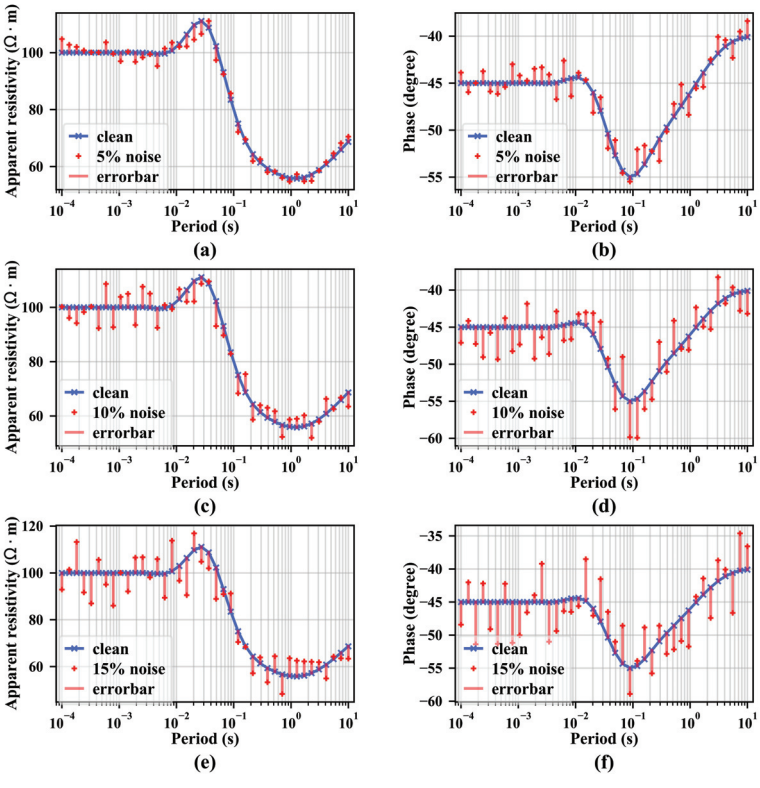


Figure 11. Supposed MT responses of a five-layer geoelectric model with different levels of noise. (**a**,**b**) represent the MT responses when the noise level is 5%. (**c**,**d**) represent the MT responses when the noise level is 10%. (**e**,**f**) represent the MT responses when the noise level is 15%. (**a**,**c**,**e**) represent the apparent resistivity responses. (**b**,**d**,**f**) represent the apparent resistivity responses. The blue lines represent the clean supposed responses. The red dots indicate the noisy responses. The red lines are the error bars between the noisy data and clean data.

From the comparison of the MT responses (Figure 12b,c), the predicted response for 5% noise has a small deviation from the supposed value, and the two types of responses are basically consistent. A 10% noise level will increase the deviation, and 15% noise will cause the most serious deviation. In particular, with the deviation of the apparent resistivity response, 15% noise causes the deviation to cover almost the whole frequency band. For the phase responses, the deviation caused by 15% noise covers most of the frequency range, from low to intermediate frequencies.

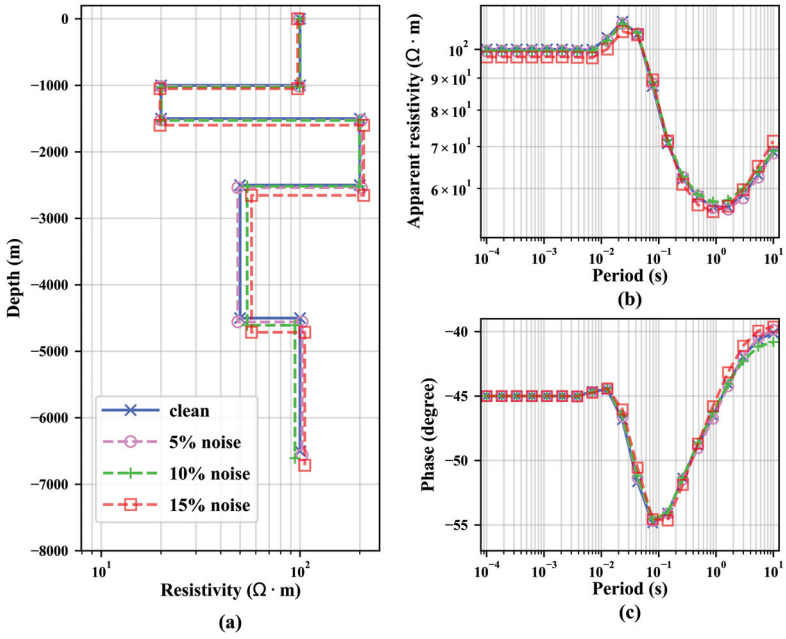


Figure 12. The predicted five-layer geoelectric models and their MT responses under noisy conditions. (**a**–**c**) represent the geoelectric model, apparent resistivity responses and phase responses, respectively. The blue lines represent the supposed three-layer electrical model and clean MT responses, and the purple lines represent the predicted geoelectric model and its MT responses when the noise level is 5%. The green lines represent the predicted geoelectric model and its MT responses when the noise level is 10%. The red lines represent the predicted geoelectric model and its MT responses when the noise level is 15%.

Table 4. Comparison of the accuracy of the predicted five-layer geoelectric model under different noise levels.

		$\rho(\Omega\cdot\mathbf{m})$					$h(\mathbf{m})$				Fitness
		ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	$h_1(\mathbf{m})$	$h_2(\mathbf{m})$	$h_3(\mathbf{m})$	$h_4(\mathbf{m})$	
Supposed mode		100.00	20.00	200.00	50.00	100.00	1000.00	500.00	1000.00	2000.00	
5% noise	model	99.47	19.98	203.61	48.63	102.5	1016.27	505.19	1017	2020.14	0.0241
	misfit [1]	0.53	0.12	1.81	2.73	2.5	1.63	1.038	1.7	1.01	
10% noise	model	99.54	20.09	199.34	54.21	94.39	1025.08	502.03	989.19	2091.82	0.0491
	misfit	0.46	0.44	0.33	8.41	5.61	2.51	0.41	1.08	4.59	
15% noise	model	97.28	19.77	209.19	57.04	105.71	1050.71	549.93	1054.34	2058.1	0.077
	misfit	2.71	1.13	4.6	14.08	5.71	5.07	9.99	5.43	2.91	

[1] The misfit = $|v_{pred} - v_{supp}|/v_{supp}$, v_{pred} is the predictive value, v_{supp} is the parameter of the supposed model.

5.2. Real Application Data

The COPROD2 dataset is a public dataset for testing the MT inversion effect, which contains measured MT response data [44]. However, the underground structure is not accurately proven. We can evaluate the inversion effect by observing the difference in fit between the predicted responses and the measured responses. The prediction effect of the proposed memetic strategy is compared with that of the traditional PSO method. The COPROD2 data contain 35 observation points. We selected the fifth, tenth, 15th and 20th observation points as our test data. The measured data can be divided into YX mode

and XY mode according to the polarization mode. The prediction results under the two modes are shown in Figures 13 and 14.

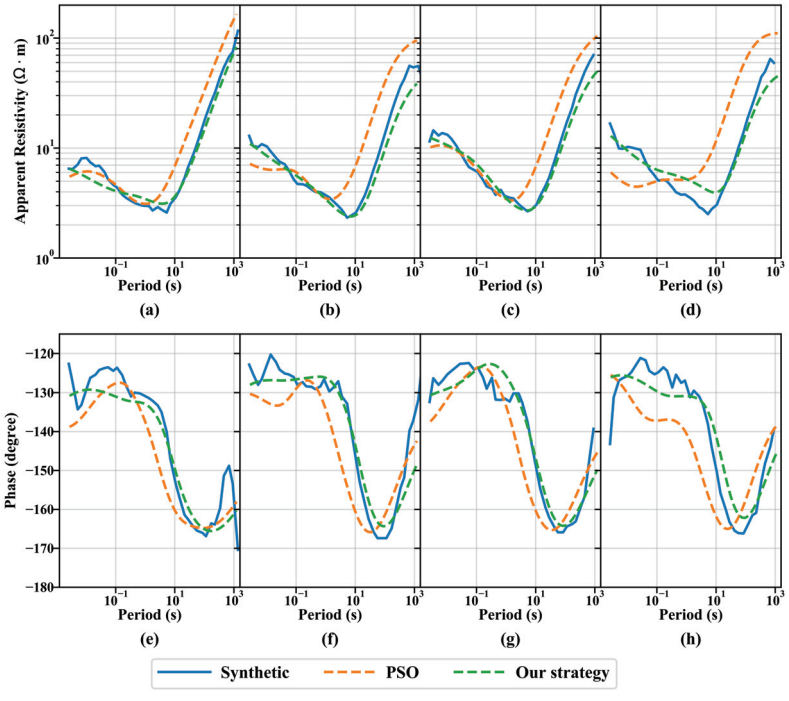


Figure 13. Comparison of the predicted and measured responses in YX mode. (**a**–**d**) represent the apparent resistivity response curves, and (**e**–**h**) represent the phase response curves. (**a**,**e**) represent the response curves of the fifth observation station, (**b**,**f**) represent the response curves of the tenth observation station, (**c**,**g**) represent the response curves of the 15th observation station, and (**d**,**h**) represent the response curves of the 20th observation station. The blue lines represent the measured response curves. The yellow lines represent the response curves predicted by traditional PSO. The yellow lines represent the response curves predicted by the proposed memetic strategy.

Among the results for the four observation stations, our memetic strategy prediction results are significantly better than the traditional PSO prediction results. When the measured responses fluctuate gently, the predicted responses can fit the measured responses. However, the prediction results of traditional PSO are consistent only with the measured responses in the change trend, and the predicted value has a large deviation. When the measured responses fluctuate violently, the predicted results of the two methods are quite different from the measured response.

Considering the volume effect of electromagnetic waves and the static effect near the surface, the response curve of the one-dimensional geoelectric model has difficulty matching the measured curve perfectly. In addition, the violent fluctuations in the measured data are mainly concentrated in the high-frequency range, which is also influenced by human noise, magnetic storms and substation interference. This nonrandom noise increases the difficulty of inversions.

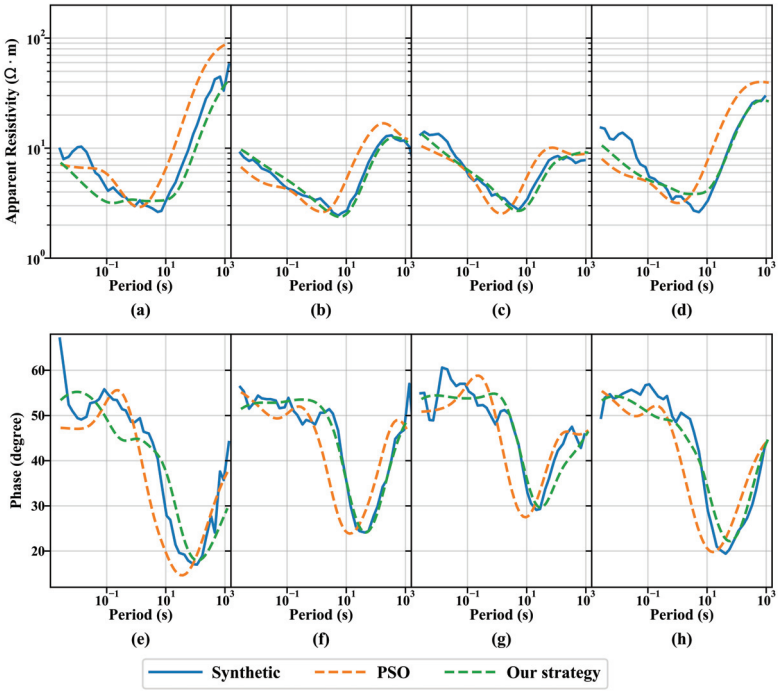


Figure 14. Comparison of the predicted and measured responses in XY mode. (**a**–**d**) represent the apparent resistivity response curves, and (**e**–**h**) represent the phase response curves. (**a**,**e**) represent the response curves of the fifth observation station, (**b**,**f**) represent the response curves of the tenth observation station, (**c**,**g**) represent the response curves of the 15th observation station, and (**d**,**h**) represent the response curves of the 20th observation station. The blue lines represent the measured response curves. The yellow lines represent the response curves predicted by traditional PSO. The yellow lines represent the response curves predicted by the proposed memetic strategy.

6. Conclusions

For MT inversions, we propose a memetic strategy on the basis of traditional PSO, which includes four parts: opposition-based learning, dynamic inertia weights, sine-cosine acceleration coefficients and gene mutation. The test results of the different models show that reverse learning can selectively enhance the population diversity and accelerate the optimization process. The dynamic inertia weights based on sine mapping can strengthen the optimization ability by fusing previous cognitive experience. By balancing the influence of individual cognition and group cognition on the evolutionary process, the sine-cosine acceleration coefficients can improve the global optimization capability in the early stages of the optimization process and maintain convergence stability in the later stages. Genetic mutation can further strengthen the ability to find the best solution by enhancing the population diversity.

The noise test verifies that this memetic strategy can improve the noise immunity of PSO. Moreover, the proposed strategy outperforms the traditional PSO method on the measured MT data. We have greatly improved the ability of PSO to invert MT data by enhancing the diversity of the population and fusing the individual and social cognition of the population.

Author Contributions: Conceptualization, R.L. and L.G.; methodology, L.G.; software, N.Y.; validation, L.G., R.L. and E.W.; formal analysis, E.W. and J.L.; data curation, X.F.; writing—original draft preparation, R.L.; writing—review and editing, Y.L. and J.L.; funding acquisition, N.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was jointly supported in part by the National Natural Science Foundation of China (42074081 and 41974158), the Postdoctoral Science Foundation of Chongqing under Grant (cstc2019jcyj-bshX0105 and cstc2020jcyj-bshX0115), and the Fund from the Key Laboratory of Geophysical Electromagnetic Probing Technologies of Ministry of Natural Resources (KLGEPT202002).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Seillé, H.; Visser, G. Bayesian inversion of magnetotelluric data considering dimensionality discrepancies. *Geophys. J. Int.* **2020**, *223*, 1565–1583. [CrossRef]
2. Howard, A.Q. Special Issue on Electromagnetic Methods in Applied Geophysics—Introduction. *IEEE Trans. Geosci. Remote Sens.* **1984**, *22*, 2.
3. Zhdanov, M. *Foundations of Geophysical Electromagnetic Theory and Methods*; Elsevier: Oxford, UK, 2009.
4. Ramananjaona, C.; MacGregor, L.; Andréis, D. Sensitivity and inversion of marine electromagnetic data in a vertically anisotropic stratified earth. *Geophys. Prospect.* **2011**, *59*, 341–360. [CrossRef]
5. Kordy, M.; Wannamaker, P.; Maris, V.; Cherkaev, E.; Hill, G. 3-D magnetotelluric inversion including topography using deformed hexahedral edge finite elements and direct solvers parallelized on SMP computers—Part I: Forward problem and parameter Jacobians. *Geophys. J. Int.* **2015**, *204*, 74–93. [CrossRef]
6. Constable, S.C.; Parker, R.L.; Constable, C.G. Occam's inversion: A practical algorithm for generating smooth model from electromagnetic sounding data. *Geophysics* **1987**, *52*, 289–300. [CrossRef]
7. Li, R.; Hu, X.; Xu, D.; Liu, Y.; Yu, N. Characterizing the 3D hydrogeological structure of a debris landslide using the transient electromagnetic method. *J. Appl. Geophys.* **2020**, *175*, 103991. [CrossRef]
8. Conway, D.; Alexander, B.; King, M.; Heinson, G.; Kee, Y. Inverting magnetotelluric responses in a three-dimensional earth using fast forward approximations based on artificial neural networks. *Comput. Geosci.* **2019**, *127*, 44–52. [CrossRef]
9. Grana, D.; Fjeldstad, T.; Omre, H. Bayesian Gaussian Mixture Linear Inversion for Geophysical Inverse Problems. *Math. Geosci.* **2017**, *49*, 493–515. [CrossRef]
10. Wang, G.G.; Tan, Y. Improving Metaheuristic Algorithms With Information Feedback Models. *IEEE Trans. Cybern.* **2019**, *49*, 542–555. [CrossRef]
11. Wang, G.G.; Guo, L.; Gandomi, A.H.; Hao, G.S.; Wang, H. Chaotic Krill Herd algorithm. *Inf. Sci.* **2014**, *274*, 17–34. [CrossRef]
12. Sharma, S.P.; Biswas, A. Global nonlinear optimization for the estimation of static shift and interpretation of 1-D magnetotelluric sounding data. *Ann. Geophys.* **2011**, *54*, 249–264.
13. Xiang, E.; Guo, R.; Liu, J.; Ren, Z.; Dong, H. Efficient hierarchical trans-dimensional Bayesian inversion of magnetotelluric data. *Geophys. J. Int.* **2018**, *213*, 3. [CrossRef]
14. Batista, J.D.C.; Sampaio, E.E.S. Magnetotelluric inversion of one- and two-dimensional synthetic data based on hybrid genetic algorithms. *Acta Geophys.* **2019**, *67*, 1365–1377. [CrossRef]
15. Pace, F.; Santilano, A.; Godio, A. Particle Swarm Optimization of 2D Magnetotelluric data. *Geophysics* **2019**, *84*, 1–59. [CrossRef]
16. Shi, Y.; Obaiahnahatti, B. A Modified Particle Swarm Optimizer. In Proceedings of the IEEE Conference on Evolutionary Computation, Anchorage, AK, USA, 4–9 May 1998; Volume 6, pp. 69–73.
17. Jordehi, A.R. Time varying acceleration coefficients particle swarm optimisation (TVACPSO): A new optimisation algorithm for estimating parameters of PV cells and modules. *Energy Conv. Manag.* **2016**, *129*, 262–274. [CrossRef]
18. Ali, A.F.; Tawhid, M.A. A hybrid particle swarm optimization and genetic algorithm with population partitioning for large scale optimization problems. *Ain Shams Eng. J.* **2017**, *8*, 191–206. [CrossRef]
19. Fernandes, I.F.; Silva, I.R.D.M.; Goldbarg, E.F.G.; Monteiro, S.M.D.; Goldbarg, M.C. A PSO-inspired architecture to hybridise multi-objective metaheuristics. *Memet. Comput.* **2020**, *12*, 235–249. [CrossRef]
20. Zhang, Z.; Wong, W.K.; Tan, K.C. Competitive swarm optimizer with mutated agents for finding optimal designs for nonlinear regression models with multiple interacting factors. *Memet. Comput.* **2020**, *12*, 219–233. [CrossRef]
21. Commer, M.; Newman, G.A. New advances in three-dimensional controlled-source electromagnetic inversion. *Geophys. J. Int.* **2008**, *172*, 513–535. [CrossRef]
22. Yang, D.; Oldenburg, D.W.; Haber, E. 3-D inversion of airborne electromagnetic data parallelized and accelerated by local mesh and adaptive soundings. *Geophys. J. Int.* **2014**, *196*, 582. [CrossRef]

23. Zhdanov, M.S.; Uma, M.; Wilson, G.A.; Velikhov, E.P.; Black, N.; Gribenko, A.V. Iterative electromagnetic migration for 3D inversion of marine controlled: Ource electromagnetic data. *Geophys. Prospect.* **2011**, *59*, 1101–1113. [CrossRef]
24. Siripunvaraporn, W. Three-dimensional magnetotelluric inversion: An introductory guide for developers and users. *Surv. Geophys.* **2012**, *33*, 5–27. [CrossRef]
25. Li, R.; Hu, X.; Li, J. Pseudo-3D constrained inversion of transient electromagnetic data for a polarizable SMS hydrothermal system in the Deep Sea. *Stud. Geophys. Geod.* **2018**, *62*, 512–533. [CrossRef]
26. Groothedlin, C.D.D.; Constable, S.C. Occam's inversion to generate smooth, two-dimensional models from magnetotelluric data. *Geophysics* **1990**, *55*, 1613–1624. [CrossRef]
27. Sasaki, Y. Full 3-D inversion of electromagnetic data on PC. *J. Appl. Geophys.* **2001**, *46*, 45–54. [CrossRef]
28. Thakur, P.; Srivastava, D.C.; Gupta, P.K. The genetic algorithm: A robust method for stress inversion. *J. Struct. Geol.* **2017**, *94*, 227–239. [CrossRef]
29. Neri, F.; Cotta, C. Memetic algorithms and memetic computing optimization: A literature review. *Swarm Evol. Comput.* **2012**, *2*, 1–14. [CrossRef]
30. Wang, D.; Tan, D.; Liu, L. Particle swarm optimization algorithm: An overview. *Soft Comput.* **2018**, *22*, 387–408. [CrossRef]
31. Liu, B.F.; Chen, H.M.; Chen, J.H.; Hwang, S.F.; Ho, S.Y. MeSwarm: Memetic particle swarm optimization. In Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2005, Washington, DC, USA, 25–29 June 2005.
32. Chiam, S.; Tan, K.; Al-Mamun, A. A memetic model of evolutionary PSO for computational finance applications. *Expert Syst. Appl.* **2009**, *36*, 3695–3711. [CrossRef]
33. Gou, J.; Lei, Y.X.; Guo, W.P.; Wang, C.; Cai, Y.Q.; Luo, W. A novel improved particle swarm optimization algorithm based on individual difference evolution. *Appl. Soft. Comput.* **2017**, *57*, 468–481. [CrossRef]
34. Clerc, M.; Kennedy, J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.* **2002**, *6*, 58–73. [CrossRef]
35. Chauhan, P.; Deep, K.; Pant, M. Novel inertia weight strategies for particle swarm optimization. *Memet. Comput.* **2013**, *5*, 229–251. [CrossRef]
36. Zhou, Y.; Hao, J.K.; Duval, B. Opposition-based Memetic Search for the Maximum Diversity Problem. *IEEE Trans. Evol. Comput.* **2017**, *21*, 731–745. [CrossRef]
37. Ma, X.; Zhang, Q.; Tian, G.; Yang, J.; Zhu, Z. On Tchebycheff decomposition approaches for multiobjective evolutionary optimization. *IEEE Trans. Evol. Comput.* **2017**, *22*, 226–244. [CrossRef]
38. Tizhoosh, H.R. Opposition-based learning: A new scheme for machine intelligence. In Proceeding of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), Vienna, Austria, 28–30 November 2005; Volume 1, pp. 695–701.
39. Wang, W.; Wang, H.; Sun, H.; Rahnamayan, S. Using opposition-based learning to enhance differential evolution: A comparative study. In Proceeding of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 71–77.
40. Ma, X.; Liu, F.; Qi, Y.; Gong, M.; Yin, M.; Li, L.; Jiao, L.; Wu, J. MOEA/D with opposition-based learning for multiobjective optimization problem. *Neurocomputing* **2014**, *146*, 48–64. [CrossRef]
41. Bansal, J.C.; Singh, P.; Saraswat, M.; Verma, A.; Jadon, S.S.; Abraham, A. Inertia weight strategies in particle swarm optimization. In Proceeding of the 2011 Third World Congress on Nature and Biologically Inspired Computing, Salamanca, Spain, 19–21 October 2011; IEEE: New York, NY, USA, 2011; pp. 633–640.
42. Chen, K.; Zhou, F.; Yin, L.; Wang, S.; Wang, Y.; Wan, F. A hybrid particle swarm optimizer with sine cosine acceleration coefficients. *Inf. Sci.* **2018**, *422*, 218–241. [CrossRef]
43. Ibrahim, A.O.; Shamsuddin, S.M.; Abraham, A.; Qasem, S.N. Adaptive memetic method of multi-objective genetic evolutionary algorithm for backpropagation neural network. *Neural Comput. Appl.* **2019**, *31*, 4945–4962. [CrossRef]
44. Beusekom, A.; Parker, R.; Bank, R.; Gill, P.; Constable, S. The 2-D magnetotelluric inverse problem solved with optimization. *Geophys. J. Int.* **2011**, *184*, 639–650. [CrossRef]

mathematics

MDPI

Article

Quantum-Inspired Differential Evolution with Grey Wolf Optimizer for 0-1 Knapsack Problem

Yule Wang and Wanliang Wang *

College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China; 1111712014@zjut.edu.cn
* Correspondence: zjutwwl@zjut.edu.cn

Abstract: The knapsack problem is one of the most widely researched NP-complete combinatorial optimization problems and has numerous practical applications. This paper proposes a quantum-inspired differential evolution algorithm with grey wolf optimizer (QDGWO) to enhance the diversity and convergence performance and improve the performance in high-dimensional cases for 0-1 knapsack problems. The proposed algorithm adopts quantum computing principles such as quantum superposition states and quantum gates. It also uses adaptive mutation operations of differential evolution, crossover operations of differential evolution, and quantum observation to generate new solutions as trial individuals. Selection operations are used to determine the better solutions between the stored individuals and the trial individuals created by mutation and crossover operations. In the event that the trial individuals are worse than the current individuals, the adaptive grey wolf optimizer and quantum rotation gate are used to preserve the diversity of the population as well as speed up the search for the global optimal solution. The experimental results for 0-1 knapsack problems confirm the advantages of QDGWO with the effectiveness and global search capability for knapsack problems, especially for high-dimensional situations.

Keywords: quantum computing; differential evolution; grey wolf optimizer; evolutionary algorithm; 0-1 knapsack problem

Citation: Wang, Y.; Wang, W. Quantum-Inspired Differential Evolution with Grey Wolf Optimizer for 0-1 Knapsack Problem. *Mathematics* **2021**, *9*, 1233. https://doi.org/10.3390/math9111233

Academic Editor: Amir H. Alavi

Received: 1 March 2021
Accepted: 5 May 2021
Published: 28 May 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

1. Introduction

The 0-1 knapsack problem (KP01) is a classical combinatorial optimization problem. It has many practical applications, such as project selection, investment decisions, and complexity theory [1,2]. Two classes of approaches were previously proposed to solve the KP01 [3]. The first class of approaches includes exact methods based on mathematical programming and operational research. It is possible to obtain the exact solutions of small-scale KP01 problems by exact methods such as branching and bound algorithm [4] and dynamic programming [5]. However, KP01 problems in various complex situations are NP-hard problems, and it is impractical to obtain optimal solutions using deterministic optimization methods for large-scale problems. The second class contains approximate methods based on metaheuristic algorithms [6]. Metaheuristic algorithms are shown to be effective approaches to solving complex engineering problems in a reasonable time when compared with exact methods [7]. Therefore, the application of metaheuristic algorithms has drawn a great deal of attention in the field of optimization.

In recent years, most of the metaheuristic algorithms, such as the genetic algorithm (GA) [8], ant colony optimization (ACO) [9], particle swarm algorithm (PSO) [10], artificial bee colony (ABC) [11], cuckoo search (CS) [12], firefly algorithm (FA) [13], and improved approaches based on these algorithms [14–19], were applied to KP01 problems and achieved outstanding results. However, these metaheuristic algorithms require not only a large amount of memory for storing the population of solutions, but also a long computational time for finding the optimal solutions. In the last few years, novel metaheuristic algorithms were proposed. Wang et al. [20,21] improved the swarm intelligence optimization approach

inspired by the herding behavior of krill and came up with the krill herd (KH) algorithm to solve combinatorial optimization problems. Faramarzi et al. [22] presented a metaheuristic called the Marine Predators Algorithm (MPA) with an application in engineering design problems. MPA follows the rules that naturally govern in optimal foraging strategy and encounters a rate policy between predator and prey in marine ecosystems. Inspired by the phototaxis and Lévy flights of moths, Wang et al. [23] developed a new metaheuristic algorithm called the moth search (MS) algorithm. MS was applied to solve discounted 0-1 knapsack problems [24] and set-union knapsack problems [25]. Gao et al. [26] presented a novel selection mechanism augmenting the generic DE algorithm (NSODE) to achieve better optimization results for solving fuzzy job-shop scheduling problems. Abualigah et al. [27] proposed the arithmetic optimization algorithm (AOA) based on the distribution behavior of the main arithmetic operators in mathematics: multiplication, division, subtraction, and addition. Wang et al. [28] proposed a new nature-inspired metaheuristic algorithm called monarch butterfly optimization (MBO) by simulating the migration of monarch butterflies. MBO was applied to solve classic KP01 [29], discounted KP01 [30], and large-scale KP01 [31,32] problems with superior searching accuracy, convergent capability, and stability. In most metaheuristic algorithms, it is difficult to use the information from individuals in previous iterations in the updating process. Wang et al. [33] presented a method for reusing the information available from previous individuals and feeding previous useful information back into the updating process in order to guide later searches.

The emergence of quantum computing [34,35] was derived from the principles of quantum theory such as quantum superposition, quantum entanglement, quantum interference, and quantum collapse. Quantum computing brings new ideas to optimization due to its underlying concepts, along with the ability to process huge numbers of quantum states simultaneously in parallel. The merging of metaheuristic optimization and quantum computing recently became a growing theoretical and practical interest aiming at deriving benefits from quantum computing capabilities to enhance the convergence and speed of metaheuristic algorithms. Several scholars investigated the effect of introducing quantum computing in metaheuristic algorithms to maintain a balance between exploration and exploitation. Han and Kim proposed a genetic quantum algorithm (GQA) [36] and a quantum inspired evolutionary algorithm (QEA) [37] by merging classical evolutionary algorithms with quantum computing concepts such as the quantum bit and the quantum rotation gate. Talbi et al. [38] proposed a new algorithm inspired by genetic algorithms and quantum computing for solving the traveling salesman problem (TSP). Chang et al. [39] proposed a quantum-inspired electromagnetism-like mechanism (QEM) to solve the KP01. Xiong et al. [40] presented an analysis of quantum rotation gates in quantum-inspired evolutionary algorithms. To avoid the problem of premature convergence, mutation operation [41], crossover operation [42], new termination criterion, and new rotation gate [43] were applied to the QEA and subsequently improved.

The differential evolution (DE) algorithm [44], proposed by Storn and Price, was derived from differential vectors of solutions for global optimization. Several simple operations including mutation, crossover, and selection were used in the DE algorithm to explore the search space. Subsequently, several algorithms combining the DE algorithm with quantum computing were designed to increase global search ability. Hota and Pat [45] extended the concept of differential operators with adaptive parameter control to the quantum paradigm and proposed the adaptive quantum-inspired differential evolution (AQDE) algorithm. In addition, quantum interference operation [46] and mutation operation [47] were brought into a quantum-inspired DE algorithm.

Several metaheuristic algorithms combining the QEA and DE were proven to be effective and efficient for solving the KP01. Wang et al. [48] proposed a quantum swarm evolutionary (QSE) algorithm that updated quantum angles automatically with improved PSO. Layeb [49] presented a quantum inspired harmony search algorithm (QIHSA) based on a harmony search algorithm (HSA) and quantum computing. Zouache et al. [50] proposed a merged algorithm called quantum-inspired differential evolution with particle

swarm optimization (QDEPSO) to solve the KP01. Gao et al. [51] proposed a quantum-inspired wolf pack algorithm (QWPA) with quantum rotation and quantum collapse to improve the performance of the wolf pack algorithm for the KP01.

The grey wolf optimizer (GWO) proposed by Mirjalili et al. [52] mimics the specific behavior of grey wolves based on leadership hierarchy in nature. Srikanth et al. [53] presented a quantum-inspired binary grey wolf optimizer (QIBGWO) to solve the problem of unit commitment scheduling.

Population diversity is crucial in evolutionary algorithms to enable global exploration and to avoid poor performance due to premature convergence [54]. However, it is hard for classical algorithms to enhance diversity and convergence performance because their population quickly converges to a specific region of the solution space. In addition, these algorithms require a large amount of memory as well as long computational time to find the optimal solution for high-dimensional situations. To avoid these difficulties, we propose a new algorithm, called quantum-inspired differential evolution algorithm with grey wolf optimizer (QDGWO). The proposed algorithm combines the features of the QEA, DE, and GWO to solve the 0-1 knapsack problem. To preserve diversity throughout the evolution, the new algorithm adopts the concepts of quantum representation and the integration of the quantum operators such as quantum measurement and quantum rotation. The adaptive operations of the DE (mutation, crossover, selection) and GWO can increase the adaptation and diversification in updating individuals. The experimental results demonstrate the competitive performance of the proposed algorithm.

The rest of the paper is organized as follows: Section 2 defines the 0-1 knapsack problem. The proposed QDGWO algorithm is presented in Section 3. The experimental results and discussion are summarized in Section 4. Conclusions and directions for future work are discussed in Section 5.

2. Related Work

2.1. Knapsack Problem

The 0-1 knapsack problem is a well-known combinatorial optimal problem that has been studied in areas such as project selection, resource distribution, and the network interdiction problem. The KP01 was demonstrated to be NP-complete [55,56]. It can be described as follows:

Given a set W of m items, $W = (x_1, x_2, x_3, \ldots, x_m)$. w_i is the weight item x_i, and p_i is the profit of x_i. C is the weight capacity of the knapsack. The objective is to find the subset X_{optimal} from set of m items that maximizes the total profit, while keeping the total weight of the selected items from exceeding C.

The 0-1 knapsack problem can be defined as:

$$\begin{aligned} &\text{Maximize } f(x) = \sum_{i=1}^{m} p_i x_i \\ &\text{s.t. } \sum_{i=1}^{m} w_i x_i \leq C, x_i \in \{0,1\}, \forall i \in \{1,2,\ldots,m\} \end{aligned} \tag{1}$$

where x_i can take either the value 1 (as selected) or the value 0 (as not selected, also called rejected).

2.2. Grey Wolf Optimizer (GWO)

The GWO algorithm [52] is inspired by the leadership hierarchy and hunting mechanism of grey wolves. To model the social order of grey wolves in the GWO, the best solution is considered the alpha (α) wolf, and the second and third best solutions are beta (β) and delta (δ) wolves, respectively. The rest of the feasible solutions are considered as omega (ω) wolves. In the GWO, α, β, and δ wolves lead the hunting, and the ω wolves go after these leading wolves when searching for the global optimal solution (target) as the prey, as shown in Figure 1.

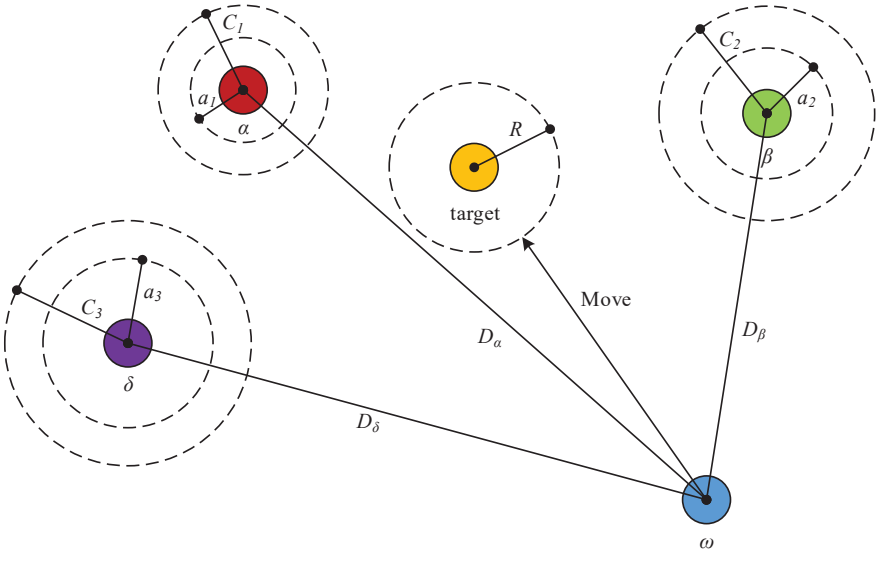


Figure 1. Position updating mechanism of GWO.

Grey wolves have the ability to recognize the location of prey and encircle them during the hunt. In order to simulate the hunting behavior of grey wolves mathematically, it is supposed that the α, β, and δ wolves have better knowledge about the potential location of prey. Therefore, the GWO saves the first three best solutions arrived at so far and forces the other ω wolves to update their positions according to the position of the best search agents.

During optimization, the GWO algorithm allows its search agents to update their position based on the location of the alpha, beta, and delta wolves with the distance vector between itself and the three best wolves when attacking the prey. Finally, the position and fitness of the alpha wolf are regarded as the global optimal solution in searching for the optimization when a termination criterion is satisfied.

3. Quantum-Inspired Differential Evolution with Adaptive Grey Wolf Optimizer

The proposed QDGWO algorithm is presented for the knapsack problems. First, the proposed algorithm adopts the quantum computing principles such as quantum representation and quantum measurement operation. Quantum representation allows the representation of the superposition of all potential states in one quantum individual. Second, adaptive mutation operations (used in the DE), crossover operations (used in the DE), and quantum observation are combined to generate new solutions as trial individuals in the solution space. Finally, the selection operator chooses the better solutions between the stored individuals and the trial individuals generated by the mutation and crossover operations of the DE. In the event that the trial individuals are worse than the current individuals, the QDGWO integrates the adaptive GWO and quantum rotation gate to preserve the diversity of the population of solutions as well as accelerate the search for the global optimum. The framework of the QDGWO algorithm is shown in Figure 2.

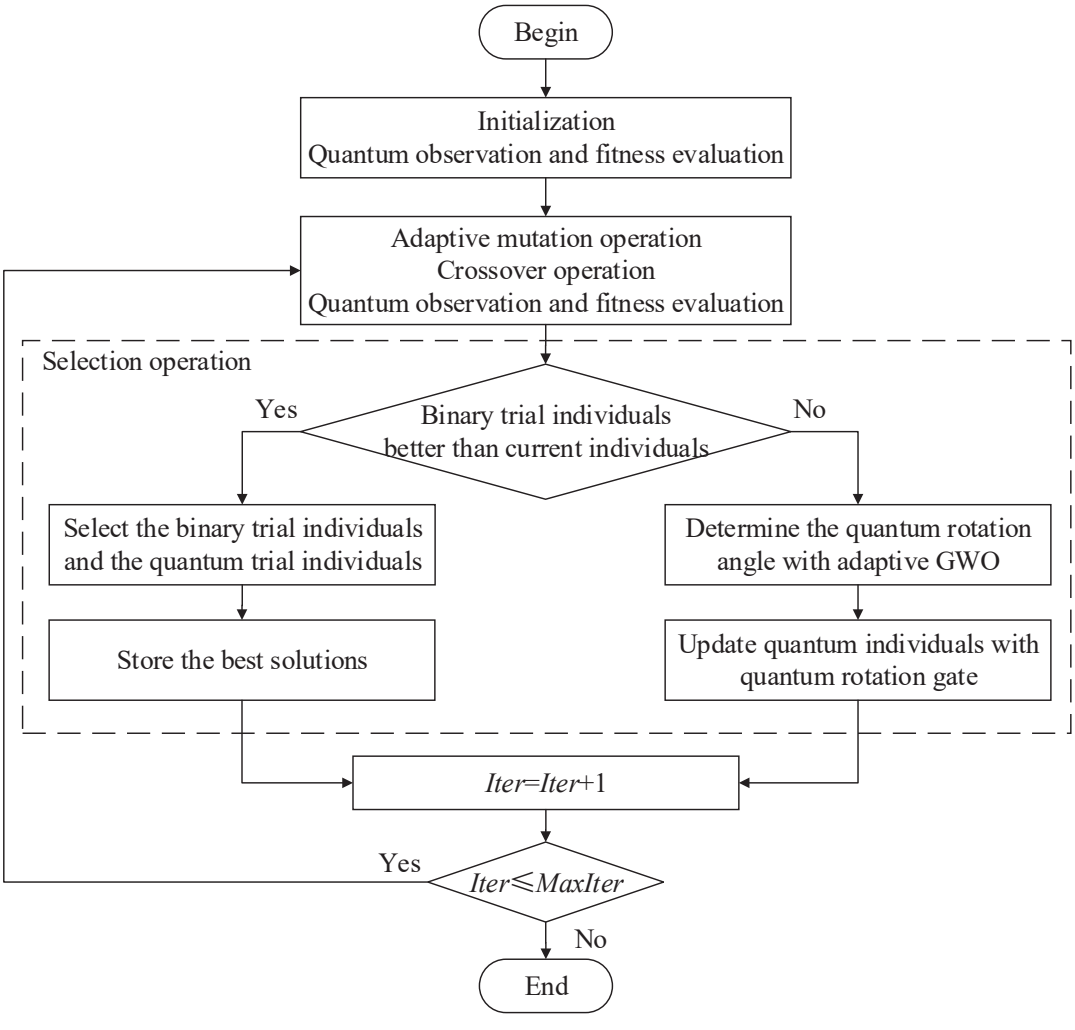


Figure 2. Framework of QDGWO.

3.1. Binary Representation

The choice of the representation for individuals, also known as individual coding, is a crucial issue in evolutionary algorithms. The proposed QDGWO algorithm adopts the binary coding, which is the most appropriate way to indicate the selection or rejection of items. Each individual X is represented as a binary vector with length m (m is the item size): $X = (x_1, x_2, \ldots, x_m)$, where m is the number of items.

$$\begin{cases} x_i = 1, & \text{if item } x_i \text{ is selected} \\ x_i = 0, & \text{if item } x_i \text{ is rejected} \end{cases} \tag{2}$$

The following example shows the binary representation for item selection: x_1 and x_3 from the item set W are selected: $W = \{x_1, x_2, x_3, x_4\} \rightarrow X = (1\ 0\ 1\ 0)$.

The binary population $P(t) = \left(X_1^t, X_2^t, \ldots, X_n^t\right)$ is made up of the binary individuals at the tth generation, where n is the size of population.

3.2. Quantum Representation

The representation of the AQDE [45] is used in the proposed algorithm, where each quantum individual q corresponds to a phase vector q_θ, which is a string of phase angles θ_i ($1 \leq i \leq m$), which can be given by

$$q_\theta = [\theta_1, \theta_2, \ldots, \theta_m], \theta_i \in [0, 2\pi] \tag{3}$$

where m is the length of the quantum bit (qubit) individual.

Each quantum individual q is a string of qubits:

$$q = \left[\begin{array}{c|c|c|c} \cos(\theta_1) & \cos(\theta_2) & \cdots & \cos(\theta_m) \\ \sin(\theta_1) & \sin(\theta_2) & & \sin(\theta_m) \end{array} \right] \tag{4}$$

The probability amplitudes of a quantum bit are expressed as a pair of numbers ($\cos(\theta_i)$, $\sin(\theta_i)$). $|\sin(\theta_i)|^2$ represents the probability of selecting item x_i, and $|\cos(\theta_i)|^2$ represents the probability of rejecting item x_i.

The quantum population $Q(t) = \left(q_1^t, q_2^t, \ldots, q_n^t\right)$ is made up of the quantum individuals at the tth generation, where n is the size of the population.

3.3. Initialization

The general principle of superposition of quantum mechanics assumes that the original state must be considered as the result of a superposition of two or more other states in an infinite number of ways [57]. Therefore, the initial quantum individual is regarded as a superposition of all possible states. For 0-1 knapsack problems, each state represents a combination of selecting or rejecting items, and the initial quantum individual is required to generate every possible combination. For this, each vector $q_{\theta i}$ is initialized by:

$$q_{\theta i}^{t=0} = \left(\left(\frac{\pi}{4}\right) \cdot r_{i1}, \ldots, \left(\frac{\pi}{4}\right) \cdot r_{im}\right) \tag{5}$$

where r_{ij} = random{1,3,5,7}, which means r_{ij} is an odd integer generated randomly in the set $r_{ij} \in \{1, 3, 5, 7\}$ while $\theta_{ij} \in \left\{\frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}\right\}$. This means that for initial individuals, $|\cos(\theta_{ij})|^2 = |\sin(\theta_{ij})|^2 = 1/2$, so that the probabilities of selecting item x_i and rejecting item x_i are equal.

The initial quantum individual $q_i^{t=0}$ which corresponds to $q_{\theta i}^{t=0}$ can be given by:

$$q_i^{t=0} = \left[\begin{array}{c|c|c|c} \cos\theta_{i1}^0 & \cos\theta_{i2}^0 & \cdots & \cos\theta_{im}^0 \\ \sin\theta_{i1}^0 & \sin\theta_{i2}^0 & & \sin\theta_{im}^0 \end{array} \right] \tag{6}$$

where m is the length of the qubit quantum individual.

$Q(0) = \left(q_1^0, q_2^0, \ldots, q_n^0\right)$ is the initial quantum population, where n is the size of the population.

3.4. Quantum Observation and Fitness Evaluation

Based on the quantum superposition principle, a quantum state is a superposition of all possibly stationary states. The quantum superposition state will collapse to a stationary state by quantum observation. In the proposed QDGWO algorithm, the quantum superposition states are represented by quantum individuals, and the stationary states are represented through binary individuals. Before evaluating the fitness of individuals, quantum observation and reparation for quantum individuals q are required to receive binary individuals X, as shown in Algorithm 1.

After quantum observation, the fitness of binary individual X is evaluated as:

$$f\left(X_i^t\right) = \sum_{i=1}^{m} p_i x_i^t \tag{7}$$

Algorithm 1 Quantum Observation and Reparation

Input: quantum individuals q
Output: binary individuals X
$x_i \leftarrow 0$ //Initialize the bits of individual X to 0.
$w_{\text{total}} \leftarrow 0$ // Initialize the total weights of the individuals to 0.
while $(totalw \leq C)$ **do**
  $i \leftarrow$ rand_i$[1, m]$ //Generate the random integer $i \in \{1, 2, \ldots, m\}$.
  if $(x_i = 0)$ **then** $r \leftarrow \text{rand}(0, 1)$
    if$(r > |\cos(\theta_i)|^2)$ **then**
      $x_i \leftarrow 1$
      $w_{\text{total}} \leftarrow w_{\text{total}} + w_i$ //Select item x_i and include the weight w_i of item x_i in the total weight w_{total}.
    end if
  end if
end while
$x_i \leftarrow 0$
$w_{\text{total}} \leftarrow w_{\text{total}} - w_i$ //The total weight w_{total} has exceeded the capacity C when the loop is ended, so item x_i needs to be extracted from the selected items for reparation.

3.5. Adaptive Mutation Operation with Dynamic Iteration Factor

The mutation operation is one of the main operations in differential evolution. In the QDGWO, DE/best/2, proposed by Price and Storn [44], is used to select parent vectors. In this strategy, the mutation vector $q_{\theta i}^{\text{M}t}$ is generated by the vector $q_{\theta\alpha}$ corresponding to the current best binary individual X_α and the difference between two different target vectors $q_{\theta r1}$ and $q_{\theta r2}$ which are randomly selected. This difference is weighted by the differentiation control factor F.

The quantum mutation vector $q_{\theta i}^{\text{M}t}$ at the tth generation can be generated by:

$$q_{\theta i}^{\text{M}t} = q_{\theta\alpha} + F^t(q_{\theta r1} - q_{\theta r2}) \tag{8}$$

where $r_1 \in \{1, 2, \ldots, m\}$ and $r_1 \neq i; r_2 \in \{1, 2, \ldots, m\}$ and $r_2 \neq i; r_1 \neq r_2$.

To improve the performance of differential operations in different phases, we propose an adaptive strategy to determine the differentiation control factor F with the iteration of evolution:

$$F^t = F_0 + F_1 \cdot 2^{\omega} \cdot \text{rand}(0, 1) \tag{9}$$

$$\omega = e^{1 - \frac{t_{\max}}{t_{\max} - t}} \tag{10}$$

where F_0 is the initial differentiation control factor, and F_1 is the adaptive f differentiation control factor. $t_{\max}$ is the maximum iteration number of the algorithm.

With this adaptive strategy, in the early stage of the iteration, the smaller t will be proposed with a larger F^t, which is beneficial for attaining good diversity of individuals for global searching. F^t will be smaller and smaller during the iteration. In the late stages, F^t is close to F_0, which aids in local searching for the global optimal solution.

The quantum mutation individual $q_i^{\text{M}t}$ corresponding to the quantum mutant vector $q_{\theta i}^{\text{M}t}$ can be obtained by:

$$q_i^{\text{M}t} = \left[\begin{array}{c|c|c|c} \cos\theta_{i1}^{\text{M}t} & \cos\theta_{i2}^{\text{M}t} & \cdots & \cos\theta_{im}^{\text{M}t} \\ \sin\theta_{i1}^{\text{M}t} & \sin\theta_{i2}^{\text{M}t} & & \sin\theta_{im}^{\text{M}t} \end{array} \right] \tag{11}$$

where m is the length of the qubit quantum individual.

The quantum mutation population $Q^{\text{M}}(t) = (q_1^{\text{M}t}, q_2^{\text{M}t}, \ldots, q_n^{\text{M}t})$ is made up of the quantum mutation individuals at the tth generation, where n is the size of the population.

3.6. Crossover Operation

The crossover operation is another main operation in differential evolution. The trial vector $q_{\theta i}^{\mathrm{C}t}$ is generated by crossover between the mutant vector $q_{\theta i}^{\mathrm{M}t}$ and the target vector $q_{\theta i}^{t}$ with a binomial crossover strategy [44].

The quantum trial vector $q_{\theta i}^{\mathrm{C}t}$ at the tth generation can be generated by:

$$q_{\theta ij}^{\mathrm{C}t} = \begin{cases} q_{\theta ij}^{\mathrm{M}t}, \text{ if } \left(rand_j(0,1) \le CR^t\right) \text{ or } (j = rnbr\_i) \\ q_{\theta ij}^{t}, \text{ if } (rand_j(0,1) > CR^t) \text{ and } (j \ne rnbr\_i) \end{cases} \tag{12}$$

where $CR \in [0,1]$ is the probability of the crossover operation which is randomly generated at tth iteration. In addition, $rnbr\_i \in \{1, m\}$ is an integer to ensure that $q_{\theta i}^{\mathrm{C}t}$ obtains at least one vector from $q_{\theta i}^{\mathrm{M}t}$.

The quantum trial individual $q_i^{\mathrm{C}t}$ corresponding to the quantum trial vector $q_{\theta i}^{\mathrm{C}t}$ can be obtained by:

$$q_i^{\mathrm{C}t} = \left[\begin{array}{c|c|c|c} \cos\theta_{i1}^{\mathrm{C}t} & \cos\theta_{i2}^{\mathrm{C}t} & \cdots & \cos\theta_{im}^{\mathrm{C}t} \\ \sin\theta_{i1}^{\mathrm{C}t} & \sin\theta_{i2}^{\mathrm{C}t} & & \sin\theta_{im}^{\mathrm{C}t} \end{array} \right] \tag{13}$$

where m is the length of the qubit quantum individual.

The quantum trial population $Q^{\mathrm{C}}(t) = \left(q_1^{\mathrm{C}t}, q_2^{\mathrm{C}t}, \ldots, q_n^{\mathrm{C}t}\right)$ is made up of the quantum trial individuals at the tth generation, where n is the size of the population.

3.7. Selection Operation

After the crossover operation, the trial quantum individuals will be transformed into binary individuals by observation and reparation, as discussed in Section 3.4. The population of trial quantum individuals $Q^{\mathrm{C}}(t)$ is transformed into a population of trial binary individuals $P^{\mathrm{C}}(t) = \left\{X_1^{\mathrm{C}t}, X_2^{\mathrm{C}t}, \ldots, X_n^{\mathrm{C}t}\right\}$.

The selection operation generates the individuals of next iteration X_i^{t+1} between the current individuals X_i^t and the trial binary individuals $X_i^{\mathrm{C}t}$, as follows:

$$X_i^{t+1} = \begin{cases} X_i^{\mathrm{C}t}, \text{ if } (f(X_i^{\mathrm{C}t}) > f(X_i^t)) \\ X_i^t, \text{ if } (f(X_i^{\mathrm{C}t}) \le f(X_i^t)) \end{cases} \tag{14}$$

The quantum individuals of the next iteration are generated by:

$$q_{\theta i}^{t+1} = \begin{cases} q_{\theta i}^{\mathrm{C}t}, \text{ if } \left(f\left(X_i^{\mathrm{C}t}\right) > f\left(X_i^t\right)\right) \\ \text{update } q_{\theta i}^{t} \text{ by } R_{\mathrm{GWO}}, \text{ if } \left(f\left(X_i^{\mathrm{C}t}\right) \le f\left(X_i^t\right)\right) \end{cases} \tag{15}$$

where R_{GWO} is a quantum rotation gate (QRG) with an adaptive GWO. R_{GWO} is presented in Section 3.8.

3.8. Quantum Rotation Gate with Adaptive GWO

The quantum rotation gate $U(\theta_i)$ is used to update the values of the qubits in a quantum individual as follows [36]:

$$U(\theta_i) = \left[\begin{array}{cc} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{array} \right] \tag{16}$$

The quantum individuals of the next iteration after quantum rotation are presented as:

$$\left[\begin{array}{c} \alpha'_i \\ \beta'_i \end{array} \right] = \left[\begin{array}{cc} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{array} \right] \left[\begin{array}{c} \alpha_i \\ \beta_i \end{array} \right] \tag{17}$$

where $\theta_i = s(\alpha_i \beta_i)\Delta\theta_i$ is the rotation angle of the QRG, and $s(\alpha_i\ \beta_i)$ is the direction signal of the rotation angle.

The polar plot of the QRG for qubits is illustrated in Figure 3, and the quantum rotation angle parameters used in [36] are shown in Table 1.

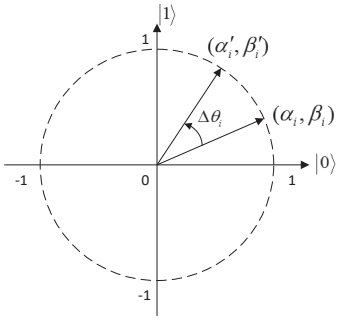


Figure 3. Polar plot of quantum rotation gate for qubit.

Table 1. Lookup table of rotation angle.

x_i	b_i	$f(x) \geq f(b)$	$\Delta\theta_i$	$s(\alpha_i\ \beta_i)$			
				$\alpha_i\ \beta_i > 0$	$\alpha_i\ \beta_i < 0$	$\alpha_i = 0$	$\beta_i = 0$
0	0	false	0	0	0	0	0
0	0	true	0	0	0	0	0
0	1	false	0	0	0	0	0
0	1	true	0.05π	-1	$+1$	± 1	0
1	0	false	0.01π	-1	$+1$	± 1	0
1	0	true	0.025π	$+1$	-1	0	± 1
1	1	false	0.005π	$+1$	-1	0	± 1
1	1	true	0.025π	$+1$	-1	0	± 1

Where $f(.)$ is the profit; $s(\alpha_i\ \beta_i)$ is the direction sign of rotation angle; and x_i and b_i are the ith bits of the binary solution x and the best solution b, respectively.

Generally speaking, the core concept of the quantum rotation gate is to motivate the probability amplitudes of each qubit in quantum individuals to converge to the corresponding bits of the current best solution in the population. Realistically, the lookup table is a convergence strategy. In this strategy, the fitness $f(x)$ of the binary solution x after quantum observation is compared with the fitness $f(b)$ of the current best solution b. The quantum rotation gate will update the probability amplitude (α_i, β_i) toward the direction that favors the emergence of the better solution between x and b. For example, if $x_i = 0$ and $b_i = 1$, and $f(x_i) > f(b_i)$, then x_i is the current best solution. This means that the state $|0\rangle$ is the optimal state for the ith qubit of a quantum individual, and that the QRG will update (α_i, β_i) to increase the probability of the state $|0\rangle$ to make the probability amplitude evolve toward the direction benefiting the appearance of $x_i = 0$. If (α_i, β_i) is located in the first quadrant as shown in Figure 2, the quantum rotation is in a clockwise direction, which favors $x_i = 0$.

Normally, quantum rotation can bring the quantum chromosomes closer to the current optimal chromosomes and generate the exploitation near the current optimal solution to find better solutions. As with other metaheuristic algorithms, individuals of the population converge more closely to the best solution after quantum rotation. The QRG makes the population evolve continuously and speeds up the convergence of the algorithm.

The magnitude of $\Delta\theta_i$ determines the granularity of the search and should be chosen appropriately. A too-small value of the rotation angle will affect the convergence speed and may even lead to a stagnant state. However, if the value is too large, the solutions may diverge or converge prematurely to a local optimal solution [37].

The traditional QRG requires predefined rotation angles, and the value and direction of θ_i should be designed for specific application problems. Because the values of quantum rotation angles are dependent on the problems, the verification of angle selection becomes important, although tedious, work when traditional QRGs are used for optimization problems. If the quantum rotation angles can be obtained adaptively without relying on predefined data, the efficiency of the QRG will be greatly improved, while the types of

applications with the QRG can be easily increased. This is exactly what metaheuristic algorithms are good at. Of the large number of metaheuristics, the GWO is a novel swarm optimization algorithm motivated by the social behavior of grey wolves. Because the GWO has the advantages of simple principles, fast searching speed, high seeking accuracy, and easy implementation, it can be easily combined with practical engineering problems [58]. Therefore, the GWO has high theoretical research value and application value, and becomes suitable for generating quantum rotation angles.

In addition, the traditional QRG motivates the probability amplitudes of each qubit in quantum individuals to converge to the corresponding bits of the current best solution in the population. If the current best solution is not the global optimal solution, the direction of quantum rotation may be far from the global optimal solution, and the algorithm may be trapped in local optimal stagnation. Since the GWO records multiple best individuals, it is useful for the QRG to jump out of local optimum with the GWO.

In the proposed QDGWO algorithm, the rotation angle of the QRG is determined with an adaptive GWO. This is described as follows: In the original GWO algorithm, the positions of other ω wolves are updated by the distance vector between themselves and the three best wolves, while for the α, β, and δ wolves, they can hunt the prey more freely. The hunting zone for α, β, and δ wolves will become smaller during the iteration. In the adaptive GWO, these features of the GWO will be inherited and developed.

For a quantum rotation gate with an adaptive GWO (R_{GWO}),$\Delta\theta_{ij}^{t}$ is calculated using the position of α, β, and δ wolves as follows:

$$\Delta\theta_{ij}^{t} = \theta \cdot \left\{ \gamma_i^{\alpha} \left(X_{\alpha j}^{t} - X_{ij}^{t} \right) + \gamma_i^{\beta} \left(X_{\beta j}^{t} - X_{ij}^{t} \right) + \gamma_i^{\delta} \left(X_{\delta j}^{t} - X_{ij}^{t} \right) \right\} \tag{18}$$

where $i \in \{1, 2, 3, \ldots, n\}$; $j \in \{1, 2, 3, \ldots, m\}$; n is the size of the population; m is the number of items; θ is the rotation angle magnitude; X_{ij}^{t} is the jth component in the binary individual of the ith wolf during tth iteration; and γ_i^{α}, γ_i^{β} and γ_i^{δ} are determined by comparing the fitness of the current binary individual with α, β, and δ wolves as follows:

$$\gamma_i^{\alpha} = \begin{cases} \frac{f(X_{\alpha}^{t})}{f(X_i^{t})}, \text{ if } f(X_i^{t}) < f(X_{\alpha}^{t}) \\ \frac{N(0,1) \times t_{\max}}{k(t_{\max} + t)}, \text{ otherwise} \end{cases} \tag{19}$$

$$\gamma_i^{\beta} = \begin{cases} \frac{f\left(X_{\beta}^{t}\right)}{f\left(X_i^{t}\right)}, \text{ if } f\left(X_i^{t}\right) < f\left(X_{\beta}^{t}\right) \\ \frac{N(0,1) \times t_{\max}}{k(t_{\max} + t)}, \text{ otherwise} \end{cases} \tag{20}$$

$$\gamma_i^{\delta} = \begin{cases} \frac{f\left(X_{\delta}^{t}\right)}{f\left(X_i^{t}\right)}, \text{ if } f\left(X_i^{t}\right) < f\left(X_{\delta}^{t}\right) \\ \frac{N(0,1) \times t_{\max}}{k(t_{\max} + t)}, \text{ otherwise} \end{cases} \tag{21}$$

where $N(0,1)$ is the Gaussian distribution, $\mu = 0$, $\sigma = 1$.

For the ω wolves, they hunt the prey based on their own positions vs. the positions of the three best wolves. It is obvious that for the ith wolf in the ω wolves group, $f\left(X_i^{t}\right) < f\left(X_{\delta}^{t}\right) \leq f\left(X_{\beta}^{t}\right) \leq f\left(X_{\alpha}^{t}\right)$. Therefore, the rotation angle of the ith wolf can be calculated with the binary individuals of α, β, δ, and the ith wolf.

For α, β, and δ wolves, they have the duty of searching for the optimal solution from their old positions. With increasing t, γ_i^{α}, γ_i^{β} and γ_i^{δ} will be much smaller in the late stages of the iteration than what they are in the early stages of the iteration, which helps the ω wolves converge toward an estimated position of prey calculated by α, β, and δ wolves. This strategy is helpful for jumping out of local optimal stagnation, especially in the final stage of iterations.

The speed of convergence and quality of the solution are greatly affected by the magnitude of the rotation angle θ, which is given by:

$$\begin{array}{l} \theta = \theta_{\min} + \left(1 - \frac{t}{t_{\max}}\right) \cdot (\theta_{\max} - \theta_{\min}) \\ 0 < \theta_{\min} < \theta_{\max} \end{array} \tag{22}$$

where θ is linearly decreasing from $\theta_{\max}$ to $\theta_{\min}$ during the iteration.

In the end, $q_{\theta i}^{t+1}$ can be generated by:

$$\theta_{ij}^{t+1} = \theta_{ij}^{t} + s_{ij}^{t} \Delta\theta_{ij}^{t} \tag{23}$$

where s_{ij}^{t} is the direction signal of the rotation angle as follows:

$$s_{ij}^{t} = \begin{cases} 1, \text{ if } \theta_{ij}^{t} \in (0, \frac{\pi}{2}) \cup (\pi, \frac{3\pi}{2}) \\ -1, \text{ if } \theta_{ij}^{t} \in (\frac{\pi}{2}, \pi) \cup (\frac{3\pi}{2}, 2\pi) \\ \pm 1, \text{ otherwise} \end{cases} \tag{24}$$

With the adaptive strategy of γ and θ, the searching granularity of the QDGWO changes from coarse to fine. A different searching granularity in the iteration will facilitate the process for search agents to reach the global optimal solution.

3.9. Procedure of QDGWO Algorithm

Based on the description above, the procedure of the QDGWO and its main steps can be summarized as shown in Algorithm 2.

Algorithm 2 QDGWO

```
t ← 0 // Initializes the iteration
Initialize Q(0) by Equations (5) and (6)
while (t < MaxIter) do
  Observe to get X(t) from q(t)//Quantum observation
  Evaluate fitness of X(t) by Equation (7)
  Apply mutation on q^M(t) by Equations (8) and (11)//Adaptive mutation
  Obtain q^C(t) by crossover by Equations (12) and (13)//Crossover
  Observe to get X^C(t) from q^C(t)//Quantum observation
  Evaluate fitness of X^C(t)
  if the trial binary individuals X^C(t) is better than X(t) then
    Update X(t+1) and q(t+1) by Equations (14) and (15)//Selection
  else
    Update q(t + 1) using QRG with adaptive GWO by Equations (16)–(24)
  end if
  t ← t + 1
end while
```

3.10. Example of QDGWO Algorithm to Solve KP01

In the following, a KP01 problem is described as an example to be solved by the QDGWO algorithm.

Given a set of ten items, $W = (x_1, x_2, x_3, \ldots, x_{10})$, and w_i is the weight of the ith item x_i; p_i is the value of x_i; and C is the weight capacity of the knapsack. Suppose that $w_i = i$, and $p_i = w_i + 5$, and $C = \frac{1}{2} \sum_{i=1}^{m} w_i = 27.5$.

Then, this 0-1 knapsack problem can be defined as:

$$\begin{aligned} &\text{Maximize } f(x) = \sum_{i=1}^{m} p_i x_i \\ &\text{s.t. } \sum_{i=1}^{m} w_i x_i \leq 27.5, x_i \in \{0,1\}, \forall i \in \{1,2,\ldots,10\} \end{aligned} \tag{25}$$

where $w_i = i$, and $p_i = w_i + 5 = i + 5$.

In this example, the population size was set to 20, and the maximum number of iterations was set to 200. The other parameters are presented in Table 2. The evolution process of q_1, an individual of the quantum population, is shown below.

Table 2. Parameters of algorithms in experiments.

	QEA	AQDE	QSE	QDGWO
Differential control parameter (F)	/	rand(0,1) × rand(0,1) × 0.1	/	$F_0 = 0.02$ $F_1 = 0.03$
Crossover control parameter (CR)	/	Gaussian distribution $N(0.5, 0.0375)$	/	Gaussian distribution $N(0.5, 0.0375)$
Parameters of PSO	/	/	$W = 0.7298$ $c_1 = 1.42$ $c_2 = 1.57$	/
Quantum rotation angle ($\Delta\theta$)	0.01π	/	/	$\theta_{\min} = 0.01\pi$ $\theta_{\max} = 0.03\pi$ $k = 10$

The initial quantum population $Q(0) = (q_1^0, q_2^0, \ldots, q_{20}^0)$ was initialized by Equations (5) and (6). The vector $q_{\theta 1}$ was initialized by $q_{\theta 1}^0 = \left(\frac{3\pi}{4}, \frac{\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}, \frac{3\pi}{4}, \frac{\pi}{4}, \frac{5\pi}{4}, \frac{\pi}{4}\right)$, and the quantum individual q_1^0 was initialized by $q_1^0 = \left[\begin{array}{c|c|c|c|c|c|c|c|c|c} \frac{-\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{array}\right]$.

After quantum observation, the binary individual X_1^0 was generated as $(1,0,1,1,0,0,1,0,0,1)$, and the fitness of X_1^0 was evaluated as $f(X_1^0) = \sum_{i=1}^{10} p_i x_i^0 = 50$.

The mutation vector $q_{\theta 1}^{M0}$ was generated by Equations (8)–(11), where t = 0; $\omega = e^{1-\frac{t_{\max}}{t_{\max}-t}} = 1$; and $F^0 = F_0 + F_1 \cdot 2^{\omega} \cdot \text{rand}(0,1) = 0.02 + 0.03 \times 2 \times 0.68 = 0.0608$ (rand(0,1) was randomly generated as 0.68).

The vector $q_{\theta\alpha}$ corresponding to the current best binary individual X_α and two different random target vectors $q_{\theta r1}$ and $q_{\theta r2}$ are shown as:

$$q_{\theta\alpha} = \left(\frac{\pi}{4}, \frac{3\pi}{4}, \frac{\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}, \frac{3\pi}{4}, \frac{\pi}{4}, \frac{\pi}{4}, \frac{3\pi}{4}, \frac{3\pi}{4}\right);$$

$$q_{\theta r1} = \left(\frac{3\pi}{4}, \frac{7\pi}{4}, \frac{7\pi}{4}, \frac{5\pi}{4}, \frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{5\pi}{4}, \frac{\pi}{4}, \frac{7\pi}{4}\right);$$

$$q_{\theta r2} = \left(\frac{7\pi}{4}, \frac{7\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{\pi}{4}, \frac{5\pi}{4}, \frac{\pi}{4}, \frac{\pi}{4}, \frac{5\pi}{4}, \frac{3\pi}{4}\right).$$

The quantum mutation vector $q_{\theta 1}^{M0}$ was generated by:

$$\begin{aligned} q_{\theta 1}^{M0} &= q_{\theta\alpha} + F^0(q_{\theta r1} - q_{\theta r2}) \\ &= (0.1892\pi, 0.75\pi, 0.3108\pi, 1.25\pi, 1.75\pi, 0.7196\pi, 0.3108\pi, 0.3108\pi, 0.6892\pi, 0.8108\pi) \end{aligned}$$

The quantum trial vector $q_{\theta 1}^{C0}$ at the tth generation was generated by Equations (12) and (13):

$$q_{\theta 1}^{C0} = (0.75\pi, 0.75\pi, 0.3108\pi, 1.75\pi, 1.25\pi, 1.75\pi, 0.3108\pi, 0.25\pi, 0.6892\pi, 0.25\pi)$$

where CR was randomly generated as 0.35, and $rnbr\_i$ = 3.

After quantum observation, the trial binary individual X_1^{C0} was generated as $(1, 0, 1, 1, 0, 0, 1, 0, 1, 0)$, and the fitness was evaluated as $f\left(X_1^{C0}\right) = \sum_{i=1}^{10} p_i x_i^0 = 49$. Because $f\left(X_1^{Ct}\right) < f\left(X_1^t\right)$, q^{t+1} should be updated by the QRG with an adaptive GWO.

The positions and profits of the current three best wolves were shown as:

$$X_\alpha^0 = (1, 1, 1, 0, 1, 1, 0, 1, 0, 0), f(X_\alpha^0) = \sum_{i=1}^{10} p_i x_i^0 = 55;$$

$$X_\beta^0 = (1, 1, 1, 1, 1, 0, 0, 0, 1, 0), f(X_\beta^0) = \sum_{i=1}^{10} p_i x_i^0 = 54;$$

$$X_\delta^0 = (0, 1, 1, 0, 0, 1, 1, 0, 1, 0), f(X_\delta^0) = \sum_{i=1}^{10} p_i x_i^0 = 52.$$

Additionally, $\theta = \theta_{\min} + \left(1 - \frac{t}{t_{\max}}\right) \cdot (\theta_{\max} - \theta_{\min}) = \theta_{\max} = 0.03\pi$; $\gamma_1^\alpha = \frac{f\left(X_\alpha^0\right)}{f\left(X_1^0\right)} = 1.1$; $\gamma_1^\beta = \frac{f(X_\beta^0)}{f(X_1^0)} = 1.08$; $\gamma_1^\delta = \frac{f(X_\delta^0)}{f(X_1^0)} = 1.04$.

Then, the quantum individual of the next iteration $q_{\theta i}^{t+1}$ was generated by Equations (16)–(24) as:

$$\begin{aligned} \Delta q_{\theta 1}^0 &= [\Delta\theta_{11}^0, \Delta\theta_{12}^0, \ldots, \Delta\theta_{110}^0] \\ &= [-0.0312\pi, 0.0966\pi, 0, -0.0642\pi, 0.0654\pi, 0.0642\pi, -0.0654\pi, 0.033\pi, 0.0636\pi, -0.0966\pi] \end{aligned}$$

$$q_{\theta 1}^0 = \left(\frac{3\pi}{4}, \frac{\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}, \frac{3\pi}{4}, \frac{\pi}{4}, \frac{5\pi}{4}, \frac{\pi}{4}\right)$$

$$\begin{aligned} q_{\theta 1}^1 &= q_{\theta 1}^0 + s_1^0 \Delta q_{\theta 1}^0 \\ &= (0.7812\pi, 0.3466\pi, 1.25\pi, 1.8142\pi, 1.3154\pi, 1.6858\pi, 0.8154\pi, 0.283\pi, 1.3136\pi, 0.1534\pi) \end{aligned}$$

The individuals of the next iteration continued to evolve until the maximum number of iterations was reached. After the iterations, the best profit of this KP01 problem was 57, and one of the best solutions was (0,1,1,1,1,1,1,0,0,0).

4. Experimental Results

To assess the performance of the proposed QDGWO algorithm, two groups of datasets are used for solving the KP01.

All experiments were conducted with Matlab 2016b, running on an Intel Core i7-4790 CPU @ 3.60 GHz, and Windows 7 Ultimate Edition.

In the first experiment described in [37], there were 50, 250, 500, 1000, 1500, 2000, 2500, and 3000 dimension sets of data by Equation (26) to test the performance of the QDGWO in high-dimension situations.

Given a set of m items, $W = (x_1, x_2, x_3, \ldots, x_m)$.

$$\begin{aligned} w_i &= \text{rand\_i}[1,\ 10] \\ p_i &= w_i + 5 \\ C &= \frac{1}{2}\sum_{i=1}^{m} w_i \end{aligned} \tag{26}$$

where w_i is the weight of the ith item x_i; p_i is the value of x_i; C is the weight capacity of the knapsack; and m is the number of items.

In the first experiment, m ranged from 50 to 3000, and the maximum number of iterations in all cases was set to 1000.

To verify the effectiveness and efficiency of the QDGWO, the results of the proposed algorithm were compared with three algorithms: QEA [37], AQDE [45], and QSE [48]. The parameters of algorithms used in the experiments are presented in Table 2, where the population size is 20. The best profits, the average profits, the worst profits, and the standard deviations of 30 independent runs are shown in Table 3 and Figures 4–7. The Wilcoxon signed-rank test [59] is performed for the results of the competing algorithms in Table 3 with a significance level $\alpha = 0.05$, where +, −, and = indicate that this algorithm is superior, inferior, or equal to the QDGWO, respectively.

Table 3. Experimental results for 0-1 knapsack problems (Experiment 1).

Number of Items		QEA	AQDE	QSE	QDGWO
50	Best	302(=)	292(−)	297(=)	302
	Average	300.63(=)	287.2(−)	294.26(−)	302
	Worst	297(=)	282(−)	290(−)	302
	Std	2.23(−)	2.97(−)	2.41(−)	0
250	Best	1517(=)	1417(−)	1446(−)	1554
	Average	1502.9(=)	1397.6(−)	1,427.7(−)	1549.3
	Worst	1496(=)	1382(−)	1412(−)	1542
	Std	4.4562(−)	7.3178(−)	8.2040(−)	2.3419
500	Best	2946(=)	2772(−)	2799(−)	3091
	Average	2917.3(−)	2732(−)	2783(−)	3072.1
	Worst	2907(−)	2717(−)	2763(−)	3058
	Std	8.8198(=)	11.3304(=)	9.2364(=)	8.9624
1000	Best	5695(−)	5382(−)	5460(−)	6121
	Average	5662.5(−)	5,364.4(−)	5442.2(−)	6085.3
	Worst	5633(−)	5342(−)	5422(−)	6048
	Std	12.7028(=)	11.2975(=)	10.1018(+)	13.4812
1500	Best	8464(−)	8198(−)	8128(−)	9126
	Average	8,439.4(−)	8,178.7(−)	8,082.8(−)	9077.1
	Worst	8414(−)	8149(−)	8039(−)	9027
	Std	15.1535(+)	13.6188(+)	20.6722(=)	21.5347
2000	Best	11,217(−)	10,951(−)	10,813(−)	12,027
	Average	11,191.2(−)	10,900.4(−)	10,781.1(−)	11,971.4
	Worst	11,164(=)	10,865(−)	10,747(−)	11,913
	Std	14.7202(+)	24.6167(=)	16.2827(+)	24.3967
2500	Best	13,907(−)	13,569(−)	13,466(−)	14,886
	Average	13,865.8(−)	13,523.3(−)	13,394.2(−)	14,831.4
	Worst	13,839(−)	13,482(−)	13,342(−)	14,751
	Std	19.0504(+)	24.5971(+)	23.5438(+)	28.4894
3000	Best	16,604(−)	16,221(−)	16,071(−)	17,769
	Average	16,549.9(−)	16,175.8(−)	16,033.2(−)	17,670.1
	Worst	16,506(−)	16,128(−)	15,995(−)	17,588
	Std	20.2286(+)	22.0621(+)	20.5269(+)	29.5280

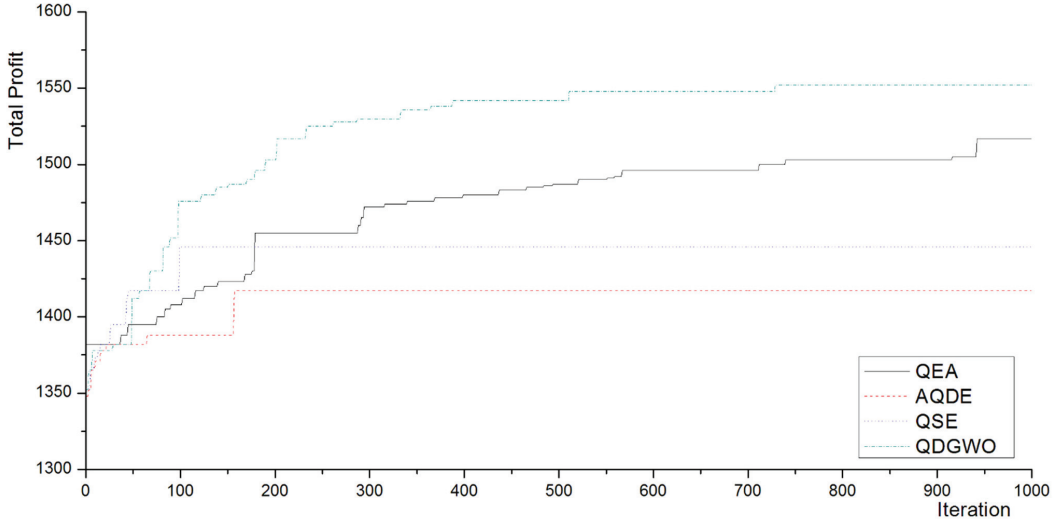


Figure 4. Best profits for the 0-1 knapsack problems (250 items in Experiment 1).

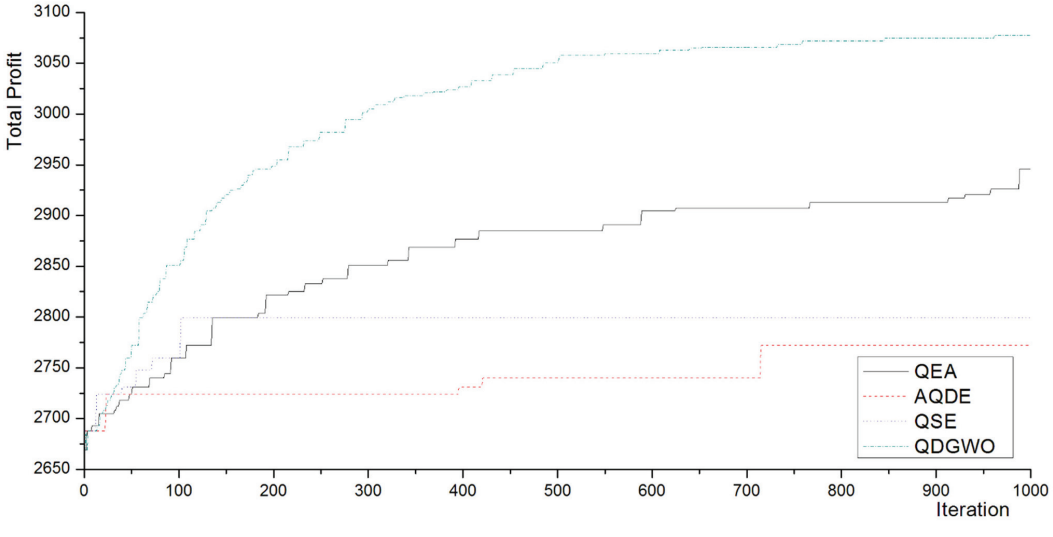


Figure 5. Best profits for the 0-1 knapsack problems (500 items in Experiment 1).

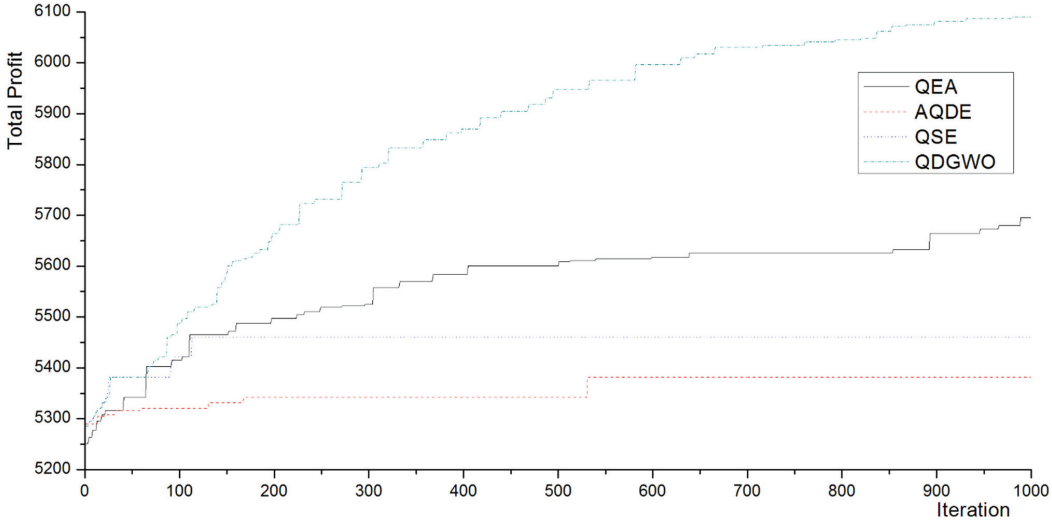


Figure 6. Best profits for the 0-1 knapsack problems (1000 items in Experiment 1).

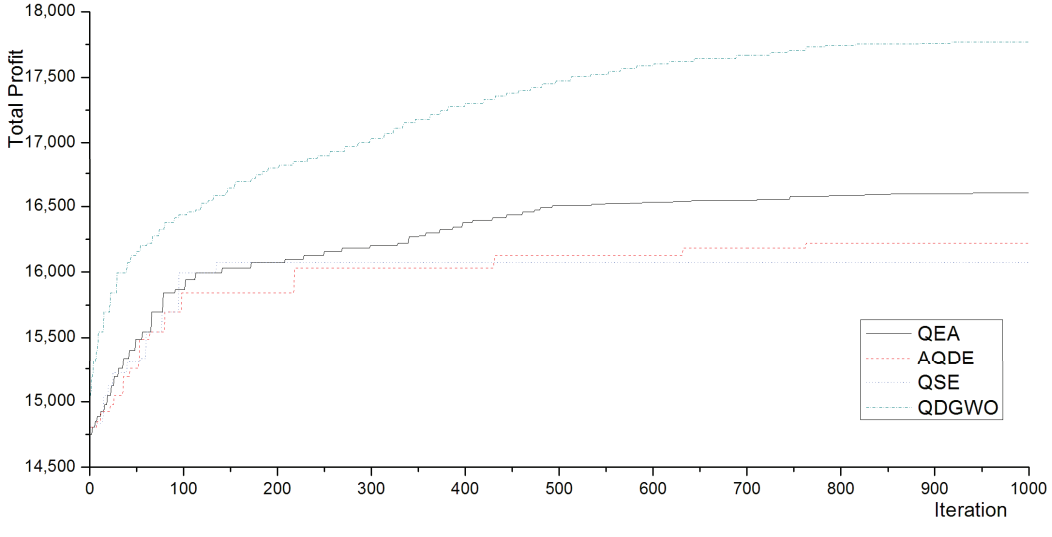


Figure 7. Best profits for the 0-1 knapsack problems (3000 items in Experiment 1).

To illustrate the importance of the role of the crossover operation of the DE in exploring the global optimum, comparative tests between the QDGWO with and without the crossover operation were performed. Moreover, we compared the binomial crossover operator of the DE with the exponential crossover operator of the DE. The best profits, the average profits, the worst profits, and the standard deviations of 30 independent runs are presented in Table 4. The Wilcoxon signed-rank test [59] is performed for the results in Table 4 with a significance level $\alpha = 0.05$, where +, −, and = indicate that this strategy is superior, inferior, or equal to the QDGWO with a binomial crossover of the DE, respectively.

Table 4. Experimental results of QDGWO algorithm without crossover of DE, with binomial crossover of DE, and with exponential crossover of DE.

Number of Items		Without Crossover of DE	With Binomial Crossover of DE (CR = 0.5)	With Exponential Crossover of DE (CR = 0.5)
500	Best	3001(−)	3046	3046(=)
	Average	2990.3(−)	3038.6	3040.1(=)
	Worst	2981(−)	3031	3031(=)
	Std	6.1752(−)	3.6191	3.4287(=)
1000	Best	5926(−)	6126	6126(=)
	Average	5893.8(−)	6109.4	6107.7(=)
	Worst	5851(−)	6096	6081(=)
	Std	18.0843(−)	7.2612	9.5447(=)
1500	Best	8752(−)	9126	9126(=)
	Average	8707.7(−)	9094.9	9090.9(=)
	Worst	8647(−)	9066	9042(=)
	Std	23.2206(−)	16.8516	21.4710(−)

In the second experiment described in [49], there were 50, 200, 500, 1000, 1500, and 2000 dimension sets of data by Equation (27) to test the performance of the QDGWO in high-dimension situations.

Given a set of m items, $W = (x_1, x_2, x_3, \ldots, x_m)$.

$$\begin{aligned} w_i &= \text{rand\_i}[1,\ 10] \\ p_i &= w_i + 5 \\ C &= \tfrac{3}{4} \sum_{i=1}^{m} w_i \end{aligned} \tag{27}$$

where w_i is the weight of the ith item x_i; p_i is the value of x_i; C is the weight capacity of the knapsack; and m is the number of items.

In the second experiment, m ranged from 50 to 2000, and the maximum number of iterations in all cases was set to 1000.

To present the performance of the proposed algorithm in the global optimization, we compared the QDGWO algorithm with the QIHSA [49] for knapsack problems. The optimization results of the success rate (SR%) and the best profit are shown in Table 5. The Wilcoxon signed-rank test [59] is performed for the results of the QIHSA in Table 5 with a significance level α = 0.05, where +, −, and = indicate that this algorithm is superior, inferior, or equal to the QDGWO, respectively.

The obtained results demonstrate the competitive performance of the proposed QDGWO algorithm. According to the results, the proposed algorithm is more efficient for the high-dimensional 0-1 knapsack problems, as shown in Table 4 and Figures 3–5. Compared with the QEA [25], AQDE [33], QSE [36], and QIHSA [37], the QDGWO was the most effective and efficient algorithm in the experiments. The advantages of the QDGWO became more obvious when the number of items was large, especially in high dimensional cases of the knapsack problems.

Table 5. Experimental results of QDGWO and QIHSA for 0-1 knapsack problems (Experiment 2).

Test	Item Size	Optimal Solution		QIHSA	QDGWO
Knapinst50	50	1177	SR% best	99.83(=) 1175(=)	100 1177
Knapinst200	200	4860	SR% best	97.83(−) 4755(−)	100 4860
Knapinst500	500	11,922	SR% best	93.74(−) 11,174(−)	98.56 11,748
Knapinst1000	1000	24,356	SR% best	87.97(−) 21,427(−)	98.14 23,903
Knapinst1500	1500	35,891	SR% best	86.31(−) 30,978(−)	97.25 34,904
Knapinst2000	2000	49,007	SR% best	85.8(−) 42,052(−)	96.36 47,223

The proposed algorithm obtains both rapid exploration and high exploitation in searching solutions. The QDGWO converges quickly to the global optimal solution. For example, the algorithm approaches the global optimum at about the 500th iteration in the case of 500 items (see Figure 4). However, the algorithm continues searching near the global optimal solution, i.e., the exploitation. To illustrate this, in the case of 500 items (see Figure 4), the QDGWO continues seeking further optimization after approaching the optimal solution and obtains better solutions in the exploitation until the end of the iterations.

Based on the results shown in Table 3, it can be concluded that the crossover operation plays a significant role in searching the solution space efficiently. However, the performance of the QDGWO is not very sensitive to which kind of crossover operator is used in the algorithm. From the experiment results, the binomial crossover operator of the DE yields slightly better optimal solutions than the exponential crossover operator of the DE in all cases. The results can be interpreted to show that quantum updating with the quantum rotation gate remains the most decisive and crucial operation in exploring the search space even if the crossover operation is required to improve the solutions.

Finally, compared with the other four methods, the experimental results show the advantages of the collaborative optimization with operations of adaptive mutation, crossover, and quantum rotation gate with the adaptive GWO in investigating the search space.

5. Conclusions

A quantum-inspired differential evolution algorithm with grey wolf optimizer (QDGWO) was proposed to solve the 0-1 knapsack problems. The proposed algorithm combined the superposition principles of quantum computing, differential evolution operations, and the hunting behaviors of grey wolves. The QDGWO used the principles of quantum computing such as quantum superposition states and quantum gates. Furthermore, it contained mutation, crossover, and selection operations of the DE. To maintain a better balance between the exploration and exploitation of searching for the global optimal solution, the proposed algorithm adapted a quantum rotation gate with the adaptive GWO to update the population of solutions. The results of tests performed for resolving the knapsack problems demonstrate that the QDGWO was able to enhance diversity and convergence performance for solving 0-1 knapsack problems. In addition, the QDGWO was effective and efficient in finding the optimal solutions for high-dimensional situations.

Although the QDGWO displays excellent performance in solving 0-1 knapsack problems, there are several directions of improvement for the proposed algorithm. First, to improve the effectiveness of the QDGWO, initial solutions of the quantum population can be generated with metaheuristic methods. In addition, the proposed approaches can be applied to solve other combinatorial optimization problems. Moreover, it is worth studying

how to use the concepts of quantum computing in other novel metaheuristic approaches such as the MPA [22] and AOA [27], as well as multi-objective optimization algorithms.

Author Contributions: Conceptualization, Y.W. and W.W.; methodology, Y.W.; software, Y.W.; validation, Y.W.; formal analysis, Y.W.; investigation, Y.W.; resources, Y.W.; data curation, Y.W.; writing—original draft preparation, Y.W.; writing—review and editing, Y.W. and W.W.; visualization, Y.W.; supervision, W.W.; funding acquisition, W.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (No. 61873240).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank the anonymous reviewers for their constructive comments and suggestions. This work was supported in part by the National Natural Science Foundation of China (No. 61873240).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kellerer, H.; Pferschy, U.; Pisinger, D. *Knapsack Problems*; Springer: Berlin/Heidelburg, Germany, 2004.
2. Wang, X.; He, Y. Evolutionary algorithms for knapsack problems. *J. Softw.* **2017**, *28*, 1–16.
3. Jourdan, L.; Basseur, M.; Talbi, E.G. Hybridizing exact methods and metaheuristics: A taxonomy. *Eur. J. Oper. Res.* **2009**, *199*, 620–629. [CrossRef]
4. Shih, W. A branch and bound method for the multiconstraint zero-one knapsack problem. *J. Oper. Res. Soc.* **1979**, *30*, 369–378. [CrossRef]
5. Toth, P. Dynamic programming algorithms for the zero-one knapsack problem. *Computing* **1980**, *25*, 29–45. [CrossRef]
6. Zou, D.; Gao, L.; Li, S.; Wu, J. Solving 0-1 knapsack problem by a novel global harmony search algorithm. *Appl. Soft Comput.* **2011**, *11*, 1556–1564. [CrossRef]
7. Fogel, D.B. Introduction to evolutionary computation. In *Evolutionary Computation 1*; Taylor & Francis Group: New York, NY, USA, 2000.
8. Chen, G.-L.; Wang, X.-F.; Zhuang, Z.-Q.; Wang, D.-S. *Genetic Algorithm and Its Applications*; The People's Posts and Telecommunications Press: Beijing, China, 2003.
9. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [CrossRef]
10. Poli, R.; Kennedy, J.; Blackwell, T. Particle swarm optimization. *Swarm Intell.* **2007**, *1*, 33–57. [CrossRef]
11. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
12. Yang, X.-S.; Deb, S. Cuckoo search via Lévy flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214.
13. Yang, X.-S.; Xin, S. Firefly algorithm, stochastic test functions and design optimisation. *Int. J. Bio-Inspired Comput.* **2010**, *2*, 78–84. [CrossRef]
14. Feng, Y.-H.; Wang, G.-G.; Wang, L. Solving randomized time-varying knapsack problems by a novel global firefly algorithm. *Eng. Comput.* **2018**, *34*, 621–635. [CrossRef]
15. Cao, J.; Yin, B.; Lu, X.; Kang, Y.; Chen, X. A modified artificial bee colony approach for the 0-1 knapsack problem. *Appl. Intell.* **2017**, *48*, 1582–1595. [CrossRef]
16. Wu, H.; Zhou, Y.; Luo, Q. Hybrid symbiotic organisms search algorithm for solving 0-1 knapsack problem. *Int. J. Bio-Inspired Comput.* **2018**, *12*, 23–53. [CrossRef]
17. Feng, Y.-H.; Jia, K.; He, Y.-C. An improved hybrid encoding cuckoo search algorithm for 0-1 knapsack problems. *Comput. Intell. Neurosci.* **2014**, *2014*, 970456. [CrossRef]
18. Zhou, Y.-Q.; Chen, X.; Zhou, G. An improved monkey algorithm for a 0-1 knapsack problem. *Appl. Soft Comput.* **2016**, *38*, 817–830. [CrossRef]
19. Sun, J.; Miao, Z.; Gong, D.-W.; Zeng, X.-J.; Li, J.-Q.; Wang, G.-G. Interval multi-objective optimization with memetic algorithms. *IEEE Trans. Cybern.* **2020**, *50*, 3444–3457. [CrossRef]
20. Wang, G.-G.; Guo, L.-H.; Gandomi, A.H.; Hao, G.-S.; Wang, H.-Q. Chaotic krill herd algorithm. *Inf. Sci.* **2014**, *274*, 17–34. [CrossRef]
21. Wang, G.-G.; Bai, D.; Gong, W.; Ren, T.; Liu, X.; Yan, X. Particle-swarm krill herd algorithm. In Proceedings of the 2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Bangkok, Thailand, 16–19 December 2018; pp. 1073–1080.

22. Faramarzi, A.; Heidarinejad, M.; Mirjalili, S.; Gandomi, A.H. Marine predators algorithm: A nature-inspired metaheuristic. *Expert Syst. Appl.* **2020**, *152*, 113377. [CrossRef]
23. Wang, G.-G. Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Memetic Comput.* **2016**, *10*, 151–164. [CrossRef]
24. Feng, Y.-H.; Wang, G.-G. Binary moth search algorithm for discounted {0-1} knapsack problem. *IEEE Access* **2018**, *6*, 10708–10719. [CrossRef]
25. Feng, Y.-H.; Yi, J.-H.; Wang, G.-G. Enhanced moth search algorithm for the set-union knapsack problems. *IEEE Access* **2019**, *7*, 173774–173785. [CrossRef]
26. Gao, D.; Wang, G.-G.; Pedrycz, W. Solving fuzzy job-shop scheduling problem using DE algorithm improved by a selection mechanism. *IEEE Trans. Fuzzy Syst.* **2020**, *28*, 3265–3275. [CrossRef]
27. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The arithmetic optimization algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
28. Wang, G.-G.; Deb, S.; Cui, Z. Monarch butterfly optimization. *Neural Comput. Appl.* **2019**, *31*, 1995–2014. [CrossRef]
29. Feng, Y.-H.; Wang, G.-G.; Deb, S.; Lu, M.; Zhao, X-J. Solving 0-1 knapsack problem by a novel binary monarch butterfly optimization. *Neural Comput. Appl.* **2017**, *28*, 1619–1634. [CrossRef]
30. Feng, Y.-H.; Wang, G.-G.; Li, W.-B.; Li, N. Multi-strategy monarch butterfly optimization algorithm for discounted {0-1} knapsack problem. *Neural Comput. Appl.* **2018**, *30*, 3019–3036. [CrossRef]
31. Feng, Y.-H.; Wang, G.-G.; Dong, J.-Y.; Wang, L. Opposition-based learning monarch butterfly optimization with Gaussian perturbation for large-scale 0-1 knapsack problem. *Comput. Electr. Eng.* **2018**, *67*, 454–468. [CrossRef]
32. Feng, Y.-H.; Yu, X.; Wang, G.-G. A novel monarch butterfly optimization with global position updating operator for large-scale 0-1 knapsack problems. *Mathematics* **2019**, *7*, 1056. [CrossRef]
33. Wang, G.-G.; Tan, Y. Improving metaheuristic algorithms with information feedback models. *IEEE Trans. Cybern.* **2019**, *49*, 542–555. [CrossRef]
34. Benioff, P. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *J. Stat. Phys.* **1980**, *22*, 563–591. [CrossRef]
35. Feynman, R.P. Simulating physics with computers. *Int. J. Theor. Phys.* **1999**, *21*, 467–488. [CrossRef]
36. Han, K.-H.; Kim, J.-H. Genetic quantum algorithm and its application to combinatorial optimization problems. In Proceedings of the International Congress on Evolutionary Computation (CEC2000), San Diego, CA, USA, 16–19 July 2000; Volume 2, pp. 1354–1360.
37. Han, K.-H.; Kim, J.-H. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Trans. Evol. Comput.* **2002**, *6*, 580–593. [CrossRef]
38. Talbi, H.; Draa, A.; Batouche, M. A new quantum-inspired genetic algorithm for solving the travelling salesman problem. In Proceedings of the 2004 IEEE International Conference on Industrial Technology, 2004, IEEE ICIT'04, Hammamet, Tunisia, 8–10 December 2004; Volume 3, pp. 1192–1197.
39. Chang, C.-C.; Chen, C.-Y.; Fan, C.-W.; Chao, H.-C.; Chou, Y.-H. Quantum-inspired electromagnetism-like mechanism for solving 0/1 knapsack problem. In Proceedings of the 2010 2nd International Conference on Information Technology Convergence and Services, Cebu, Philippines, 11–13 August 2010; pp. 1–6.
40. Xiong, H.; Wu, Z.; Fan, H.; Li, G.; Jiang, G. Quantum rotation gate in quantum-inspired evolutionary algorithm: A review, analysis and comparison study. *Swarm Evol. Comput.* **2018**, *42*, 43–57. [CrossRef]
41. Zhou, W.; Zhou, C.; Liu, G.; Lv, H.; Liang, Y. An improved quantum-inspired evolutionary algorithm for clustering gene expression data. In *Computational Methods*; Springer: Dordrecht, The Netherlands, 2006; pp. 1351–1356.
42. Xiao, J.; Yan, Y.; Lin, Y.; Yuan, L.; Zhang, J. A quantum-inspired genetic algorithm for data clustering. In Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–6 June 2008; pp. 1513–1519.
43. Han, K.-H.; Kim, J.-H. Quantum-inspired evolutionary algorithms with a new termination criterion, H$^{\varepsilon}$Gate, and two-phase scheme. *IEEE Trans. Evol. Comput.* **2004**, *8*, 156–169. [CrossRef]
44. Storn, R.; Price, K. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
45. Hota, A.R.; Pat, A. An adaptive quantum-inspired differential evolution algorithm for 0-1 knapsack problem. In Proceedings of the 2010 Second World Congress on Nature and Biologically Inspired Computing (NaBIC), Kitakyushu, Japan, 15–17 December 2010; pp. 703–708.
46. Draa, A.; Meshoul, S.; Talbi, H.; Batouche, M. A quantum-inspired differential evolution algorithm for solving the N-queens problem. *Neural Netw.* **2011**, *1*, 21–27.
47. Su, H.; Yang, Y. Quantum-inspired differential evolution for binary optimization. In Proceedings of the 2008 Fourth International Conference on Natural Computation, Jinan, China, 18–20 October 2008; Volume 1, pp. 341–346.
48. Wang, Y.; Feng, X.-Y.; Huang, Y.-X.; Pu, D.-B.; Zhou, W.-G.; Liang, Y.-C.; Zhou, C.-G. A novel quantum swarm evolutionary algorithm and its applications. *Neurocomputing* **2007**, *70*, 633–640. [CrossRef]
49. Layeb, A. A hybrid quantum inspired harmony search algorithm for 0-1 optimization problems. *J. Comput. Appl. Math.* **2013**, *253*, 14–25. [CrossRef]

50. Zouache, D.; Moussaoui, A. Quantum-inspired differential evolution with particle swarm optimization for knapsack problem. *J. Inf. Sci. Eng.* **2015**, *31*, 1757–1773.
51. Gao, Y.; Zhang, F.; Zhao, Y.; Li, C. Quantum-inspired wolf pack algorithm to solve the 0-1 knapsack problem. *Math. Probl. Eng.* **2018**, *2018*, 5327056. [CrossRef]
52. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
53. Srikanth, K.; Panwar, L.K.; Panigrahi, B.K.; Herrera-Viedma, E.; Sangaiah, A.K.; Wang, G.-G. Meta-heuristic framework: Quantum inspired binary grey wolf optimizer for unit commitment problem. *Comput. Electr. Eng.* **2018**, *70*, 243–260. [CrossRef]
54. Sudholt, D. The benefits of population diversity in evolutionary algorithms: A survey of rigorous runtime analyses. In *Theory of Evolutionary Computation*; Springer: Cham, Switzerland, 2020; pp. 359–404.
55. Pisinger, D. Where are the hard knapsack problems? *Comput. Oper. Res.* **2005**, *32*, 2271–2284. [CrossRef]
56. Vasquez, M.; Yannick, V. Improved results on the 0-1 multidimensional knapsack problem. *Eur. J. Oper. Res.* **2005**, *165*, 70–81. [CrossRef]
57. Dirac, P.A.M. *The Principles of Quantum Mechanics*, 4th ed.; Oxford University Press: New York, NY, USA, 1981; p. 12.
58. Wang, J.S.; Li, S.X. An improved grey wolf optimizer based on differential evolution and elimination mechanism. *Sci. Rep.* **2019**, *9*, 1–21. [CrossRef] [PubMed]
59. Woolson, R.F. Wilcoxon signed-rank test. In *Wiley Encyclopedia of Clinical Trials, 1–3*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2007.

mathematics

MDPI

Article

Sorting-Based Discrete Artificial Bee Colony Algorithm for Solving Fuzzy Hybrid Flow Shop Green Scheduling Problem

Mei Li [1], Gai-Ge Wang [1,2,3,*] and Helong Yu [4,*]

[1] Department of Computer Science and Technology, Ocean University of China, Qingdao 266100, China; limei@stu.ouc.edu.cn
[2] Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China
[3] Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Guangxi University for Nationalities, Nanning 530006, China
[4] College of Information Technology, Jilin Agricultural University, Changchun 130118, China
* Correspondence: wgg@ouc.edu.cn (G.-G.W.); yuhelong@jlau.edu.cn (H.Y.)

Abstract: In this era of unprecedented economic and social prosperity, problems such as energy shortages and environmental pollution are gradually coming to the fore, which seriously restrict economic and social development. In order to solve these problems, green shop scheduling, which is a key aspect of the manufacturing industry, has attracted the attention of researchers, and the widely used flow shop scheduling problem (HFSP) has become a hot topic of research. In this paper, we study the fuzzy hybrid green shop scheduling problem (FHFGSP) with fuzzy processing time, with the objective of minimizing makespan and total energy consumption. This is more in line with real-life situations. The non-linear integer programming model of FHFGSP is built by expressing job processing times as triangular fuzzy numbers (TFN) and considering the machine setup times when processing different jobs. To address the FHFGSP, a discrete artificial bee colony (DABC) algorithm based on similarity and non-dominated solution ordering is proposed, which allows individuals to explore their neighbors to different degrees in the employed bee phase according to a sequence of positions, increasing the diversity of the algorithm. During the onlooker bee phase, individuals at the front of the sequence have a higher chance of being tracked, increasing the convergence rate of the colony. In addition, a mutation strategy is proposed to prevent the population from falling into a local optimum. To verify the effectiveness of the algorithm, 400 test cases were generated, comparing the proposed strategy and the overall algorithm with each other and evaluating them using three different metrics. The experimental results show that the proposed algorithm outperforms other algorithms in terms of quantity, quality, convergence and diversity.

Keywords: green shop scheduling; fuzzy hybrid flow shop scheduling; discrete artificial bee colony algorithm; minimize makespan; minimize total energy consumption

Citation: Li, M.; Wang, G.-G.; Yu, H. Sorting-Based Discrete Artificial Bee Colony Algorithm for Solving Fuzzy Hybrid Flow Shop Green Scheduling Problem. *Mathematics* **2021**, *9*, 2250. https://doi.org/10.3390/math9182250

Academic Editor: Frank Werner

Received: 16 August 2021
Accepted: 8 September 2021
Published: 14 September 2021

1. Introduction

The growth of manufacturing has brought economic and social prosperity. Shop scheduling, as a key part of manufacturing, plays an important role in economic development. Hybrid flow shop (HFS) is a common manufacturing environment [1] that combines the features of process shop and parallel machine scheduling and is widely used in container handling [2], electronics manufacturing, chemical production, and steel production [3–5], in addition to applications in internet service architecture [6], civil engineering [7], and production planting [8]. The hybrid flow shop scheduling problem (HFSP) refers to multiple jobs to be processed in multiple stages with one or more machines in each stage, and a specific optimization objective is achieved by determining the order in which the jobs are processed and the allocation of machines to each job in each stage [1]. It is worth noting that there are two other cases of HFSP in real life [9,10]: (1) the processing time

of a job is often not fixed but fluctuates within a limited range due to worker proficiency, newness of the machine. (2) The same machine processing different jobs requires a certain setup of the machine before processing, and due to the differences between jobs, the setup time required by the machine varies from job to job. Therefore, it is more meaningful and practical to study HFSP with setup time and fuzzy job processing time.

While the world is experiencing unprecedented economic and social prosperity, environmental pollution and energy scarcity are becoming a serious problem that seriously affects the future development of humanity. In particular, the manufacturing industry takes up most of the world's energy and produces a large amount of pollutant emissions [11]. Therefore, in order to solve the energy and environmental problems, green shop scheduling, as a key aspect of manufacturing, has become a hot spot for research [12]. The purpose of green shop scheduling is to reduce energy consumption, reduce environmental pressure, and achieve sustainable development without losing economic benefits. Therefore, the widely used hybrid flow green shop scheduling problem (HFGSP) has a high research value.

However, HFGSPs that consider fuzzy job processing time are not common at present. Fu et al. [13] developed a hybrid multi-objective optimization algorithm to solve HFSP with fuzzy processing time but did not consider the energy problem. Wang et al. [14] investigated the HFGSP of job processing time variation caused by the dynamic reconfiguration process of the device to minimize the energy consumption of makespan and the whole device and proposed an improved multi-objective whale optimization algorithm to solve it.

As HFSP has a wide range of application scenarios, the uncertain job processing time meets the actual production needs and the energy saving is in line with the future direction of manufacturing. In this paper, we study the fuzzy hybrid flow green shop scheduling problem (FHFGSP) which meets the above three scenarios and is less studied currently. FHFGSP considers fuzzy job processing time and machine setup time with the objective of minimizing both makespan (MS) and total energy consumption (TEC). Uncertain completion time is denoted by triangular fuzzy numbers (TFN) and TEC is divided into three parts: machine working time, machine setup time, and machine idle time. At present, there are not many HFGSPs that consider both fuzzy processing time and work sequence-related setup time, but FHFGSP is more in line with actual production scenarios and has higher research value.

Artificial bee colony (ABC) [15] is one of the swarm intelligence algorithms, which is divided into employed bees, onlooker bees, and scout bees according to the foraging behavior of the swarm, with good global exploration and local development. ABC has been shown to be superior or close to other classical swarm intelligence algorithms [16,17]. ABC is widely used to solve shop scheduling problems [18]. To solve FHFGSP, this paper proposed a sorting-based discrete artificial bee colony algorithm (SDABC). Individuals in the population are ranked according to non-dominated solutions and similarity to the ideal solution and adopt different search and follow strategies according to the location to achieve full exploration of the solution space and discover better solutions. It is worth mentioning that SDABC can be used not only to solve FHFGSP problems such as turning shop [19]. It can also be used to solve the expansion of FHFGSP described in the first paragraph.

The main contributions of this paper are as follows:

(1) The FHFGSP with processing time fuzzy is investigated. The completion time is represented by TFN, and the energy consumption in the scheduling process is considered in three parts, which is more in line with the actual production environment.
(2) In the employed bee phase, the population was ranked based on the number of dominant solutions and the similarity of ideal solutions, and different degrees of exploration were taken for individuals according to the results of the ranking, with the best individuals being more fully explored.
(3) In the onlooker bee phase, a selection strategy is adopted so that individuals in the top ranking have a higher probability of being selected, and a mutation strategy is adopted to avoid falling into a local optimum.

The paper is organized as follows: Section 2 gives the relevant works, Section 3 describes what the FHFGSP is, gives a symbolic representation and builds a mathematical model of the FHFGSP. Section 4 details the SDABC for solving the FHFGSP. Experimental validation is presented in Section 5 and the last section contains conclusions and outlook.

2. Related Works

ABC has been successfully applied to solve shop scheduling problems due to its advantages such as few control parameters and ease of implementation [20]. As there is no research related to ABC for solving FHFGSP, this section reviews the work related to the use of ABC for solving shop scheduling problems.

Li et al. [18] proposed a novel hybrid ABC and tabu search algorithm (TABC) to solve the HFSP finite buffers, employing a TS-based adaptive neighborhood strategy that gives the TABC algorithm the ability to learn and generate neighborhood solutions in different promising regions as a means to minimize makespan. Yue et al. [21] investigated the batching and hybrid model scheduling problem in a flexible parallel production line, considering the sequence-dependent setup time between hybrid model products with the aim of minimizing the manufacturing cycle time of the line while balancing the workload between lines and maximizing the net profit. In addition, a new material availability constraint is introduced to the problem. A novel Pareto guided ABC is designed to address the current problem. Gong et al. [22] considered the impact and potential of human factors on improving productivity and reducing production costs in real production systems and proposed a hybrid ABC to solve flexible job shop scheduling problems (FJSP) with worker flexibility. Zadeh et al. [23] proposed a heuristic model based on an ABC for the dynamic FJSP. Lei et al. [24] studied the distributed unrelated parallel machine scheduling problem with preventive maintenance (DUPMSP) and proposed an ABC with division to minimize MS. Xie et al. [25] proposed an improved ABC considering machining structure evaluation to solve the flexible integrated scheduling problem of networked equipment, which is an extension of job shop scheduling. Xuan et al. [20] proposed an improved DABC with the introduction of a genetic algorithm to solve FJSP for uncorrelated parallel machines with progressively deteriorating jobs and timing dependencies.

As flow shops are very common in practical production activities, the HFSP is of high research value. Wang et al. [19] proposed a new decoding method that simultaneously considers spindle speed optimization and scheduling scheme optimization and acts on the distribution estimation algorithm to simultaneously reduce energy consumption and makspan in the turning shop. Li et al. [26] proposed an improved ABC to solve the distributed flow shop problem (DFSP) with the objective of minimizing MS. Li et al. [27] proposed a hybrid ABC to solve the parallel batch DFSP with deteriorating jobs. In the proposed algorithms, two types of problem-specific heuristics are proposed, namely batch allocation and right-shift heuristics, which can significantly shorten makespan. Gong et al. [28] proposed a hybrid multi-objective DABC for solving the blocked batch flow process shop scheduling problem with two conflicting criteria of minimizing MS and lead time. With the objective of minimizing the total process time, Pan et al. [29] solved the distributed arrangement flow job scheduling problem based on a high-performance framework of DABC. Li et al. [30] proposed an improved ABC to solve a multi-objective optimization model with the objectives of minimizing MS and processing cost for the hybrid flow shop process planning and production scheduling independently of each other. Peng et al. [31] investigated the problem of flow shop rescheduling in the actual steelmaking process, considering interruptions caused by machine failures and controllable processing times in the final stages, and proposed an improved ABC to solve the problem.

However, in actual production, the processing time of jobs is often uncertain and there is very little research on ABC solutions to fuzzy HFSP. Zhong et al. [32] proposed a new artificial swarm algorithm, the improved artificial swarm algorithm, for the multi-objective fuzzy FJSP. The objectives are to minimize the maximum fuzzy MS, maximize the weighted consistency index and minimize the maximum fuzzy machine workload.

Most of the research on the use of ABC to solve shop scheduling problems is in the area of improving economic efficiency. Very little research has been done on saving energy and reducing pollution emissions. Li et al. [33] designed an improved ABC to solve a multi-objective low-carbon job shop scheduling problem with variable machining speed constraints. Zhang et al. [34] studied HFGSP with variable machine processing speed to minimize MS and TEC and proposed a multi-objective DABC (MDABC) to solve HFGSP. However, in HFGSP, the processing time of the job is set to an exact value, which is not fully compatible with the actual production environment. In real life, the processing time of the job often deviates due to the operator's business ability, machine aging, etc. Moreover, the neighborhood search adopted by MDABC in the employed bee phase and the binary race strategy adopted in the onlooker bee phase make the algorithm suffer from the problem that it cannot fully explore in the solution space, the convergence of the algorithm is not high, and it is easy to fall into local optimum.

For this reason, this paper studies the FHFGSP with uncertain job processing time and proposes SDABC to solve FHFGSP. In SDABC, the dominant individuals guide the poor individuals to update in the employed bee phase, which improves the convergence speed of the population, and the proposed ranking-based selection strategy and mutation strategy can prevent individuals from falling into local optimum in the onlooker bee phase. FHFGSP is consistent with the actual production environment and production requirements, but it is not common in previous studies.

3. FHFGSP

This section first details the problem definition of FHFGSP, then the rules of TFN operations are explained, and finally the symbolic representation of FHFGSP is given and the mathematical model of FHFGSP is developed.

3.1. Description of the Problem

FHFGSP combines the features of fuzzy scheduling and HFSP. In FHFGSP, n jobs will be processed in m ($m \geq 2$) stages in the same order. Each stage j has at least one machine $M_{j,k}$ ($k \geq 1$) and at least one stage has multiple machines [1,35,36]. The processing time $T_{i,j,v}$ of job$_i$ on machine $M_{j,k}$ is uncertain and is given by the triple [37] ($t^o_{i,j,v}$, $t^m_{i,j,v}$, $t^p_{i,j,v}$) where $t^o_{i,j,v} \leq t^m_{i,j,v} \leq t^p_{i,j,v}$. $t^o_{i,j,v}$ denotes the optimal processing time, $t^m_{i,j,v}$ denotes the most probable processing time, and $t^p_{i,j,v}$ denotes the worst processing time.

The constraints for FHFGSP are formulated as follows:

(1) Jobs are not allowed to be interrupted and preempted when there is a job being processed on the machine, and the machine is not allowed to stop.
(2) At the beginning, all jobs and machines are available.
(3) Only one job can be processed by any one machine at any one time and any job is only allowed to be processed by one machine at any one time.
(4) Machines at the same stage process jobs at the same speed with the same power.
(5) Machines are allowed to idle.
(6) Machines can only process jobs at a selected speed. This cannot be changed during the processing.

The objective to be optimized by FHFGSP is to minimize MS and TEC. In this paper, the TEC is divided into three parts: when the machine is idle, when the machine is in the setup phase, and when the machine is processing jobs. There are three ways to reduce MS: (1) reduce machine idle time, which is influenced by the job sequence. (2) Reduce machine setup time, which also reduces TEC, which is also influenced by the job sequence. (3) Reducing the time of the job being processed, which means increasing the processing speed of the job. However, the energy consumption of the machine when processing a job is proportional to the processing speed of the job [38], and reducing the job processing time increases the TEC. Since the two objectives to be optimized are in conflict with each other,

this paper solves the FHFGSP by adjusting the job sequence and the speed of the machine when processing the job.

3.2. TFN Concepts and Operations

The concept of fuzzy sets was introduced by Zadeh [39] and the basic idea is to fuzzily the absolute affiliation in classical sets. It can be used to solve real-life uncertainty problems [40]. This subsection gives the rules for the operation of the TFN to facilitate the solution of the GFHSP.

For any two TFNs $A = (a_1, a_2, a_3)$ and $B = (b_1, b_2, b_3)$ the rules for each operation are as follows:

1. Additive operations

$$A + B = (a_1 + b_1,\ a_2 + b_2,\ a_3 + b_3) \tag{1}$$

2. Multiplication operations

$$A \times B = (a_1 \times b_1, a_2 \times b_2, a_3 \times b_3) \tag{2}$$

3. Comparative operations

$$\bar{A} = (\frac{a_1 + 2a_2 + a_3}{4}) \tag{3}$$

The TFN comparison operation is divided into three steps and has three judgement criteria.

Step 1: Get $\bar{A}$ and $\bar{B}$ by (3). If $\bar{A} >$ (<) $\bar{B}$, then $A >$ (<) B.

Step 2: If $\bar{A} = \bar{B}$, then compare a_2 and b_2. If $a_2 >$ (<) b_2, then $A >$ (<) B.

Step 3: If $\bar{A} = \bar{B}$ and $a_2 = b_2$, then compare the difference between a_3 and a_1. If $a_3 - a_1 >$ (<) $b_3 - b_1$, then $A <$ (>) B.

3.3. Mathematical Models

After understanding the basic concepts of FHFGSP and TFN, mathematical modelling of FHFGSP from the perspective of optimization objectives is needed to facilitate a better understanding of the problem to solve it. The interpretation of the relevant symbols appearing in the FHFGSP is shown in Table 1.

Objective:

$$Min\{MS, TEC\} \tag{4}$$

Subject to:

$$\sum_{k \in M_j} \sum_{v \in V_j} x_{i,j,k,v} = 1, \forall i \in I, j \in J \tag{5}$$

$$e_{i,j} - b_{i,j} = \sum_{k \in M_j} \sum_{v \in V_j} x_{i,j,k,v} \cdot pt_{i,j,v} \tag{6}$$

$$b_{i,j} - e_{i,j-1} \geq 0,\ j \in \{2, \ldots, m\} \tag{7}$$

$$z_{i,i^*,j,k} + z_{i^*,i,j,k} \leq 1,\ \ \forall i, i^* \in I, k \in M_j \tag{8}$$

$$b_{i,j} - \sum_{i^* \in I} e_{i^*,j} \cdot z_{i^*,i,j,k} - \sum_{k \in M_j} \sum_{i^* \in I} z_{i^*,i,j,k} \cdot st_{i^*,i,j} \geq 0 \tag{9}$$

$$e_{i,j} - \sum_{k \in M_j} \sum_{v \in V_j} x_{i,j,k,v} \cdot pt_{i,j,v} - \sum_{k \in M_j} \sum_{i^* \in I} z_{i^*,i,j,k} \cdot st_{i^*,i,j} = 0 \tag{10}$$

$$MS = \max e_{i,m} \tag{11}$$

$$TEC = PE + SE + IE \tag{12}$$

$$pp_{i,j,v} = p_{i,j} / c_{j,v} \tag{13}$$

$$PE = \sum_{j\in J}\sum_{k\in M_j}\sum_{i\in I}\sum_{v\in V_j} x_{i,j,k,v}\cdot T_{i,j,v}\cdot pp_{i,j,v} \tag{14}$$

$$SE = \sum_{j\in J}\sum_{k\in M_j}\sum_{i\in I}\sum_{i^*\in I}\sum_{v\in V_j} x_{i,j,k,v}\cdot z_{i^*,i,j,k}\cdot st_{i^*,i,j}\cdot sp \tag{15}$$

$$IE = \sum_{j\in J}\sum_{k\in M_j}\left[E_{k,j} - B_{k,j} - \sum_{i\in I}\sum_{v\in V_j} x_{i,j,k,v}\cdot\left(pt_{i,j,v} + \sum_{i^*\in I} z_{i^*,i,j,k}\cdot st_{i^*,i,j}\right)\right]\cdot ip \tag{16}$$

where (4) gives the objective of the FHFGSP to minimize both MS and TEC (5)–(10) give the associated constraints. (5) guarantees that each job i can be assigned to a specific machine k for processing at speed v at each stage j. (6)–(9) guarantees that no interruptions and preemptions by jobs are allowed during the processing and setup phases. (10) indicates that the machine starts processing as soon as setup is complete. (11) indicates that MS is determined by the end time of the last job to be processed in the final stage. (12) indicates that the TEC consists of three components, PE indicates the energy consumption of the machine while processing the job, SE indicates the energy consumption of the machine during the setup time, IE indicates the energy consumption of the machine during the idle time. (13) denotes the actual power of job i when it is processed at speed v in stage j. (14)–(16) are the specific information of PE, SE, and IE, respectively, all energy consumption is obtained by multiplying power by time.

Table 1. Nomenclature.

Symbol	Meaning
MS	The time required to complete the entire scheduling program
TEC	The total energy consumption required to complete the entire scheduling program
I	The set of jobs and $\|I\| = n$
i	Index of the job, indicating the i-th job
J	The set of stages and $\|J\| = m$
j	Index of the stage, indicating the j-th stage
M_j	The set of machines at stage j
k	Index of the machine
$e_{i,j}$	The ending time of job i at stage j and its value is greater than 0
$p_{i,j}$	The standard processing time for job i at stage j
$c_{j,v}$	The adjustment factor when the machine is running at speed v at stage j
$T_{i,j,v}$	The time required for job i to be processed at speed v at stage j
$st_{i^*,i,j}$	The setup time from job i^* (i^* is the previous job of i) to job i at stage j. If $i^* = i$, $s_{i^*,i,j}$ indicates the setup time required for job i as the first job
sp	Energy consumed by the machine per unit time during the setup phase
ap	Energy consumed by auxiliary equipment per unit of time throughout the scheduling process
ip	Energy consumed by the machine per unit of time during the idle phase
$b_{i,j}$	The beginning time of job i at stage j and its value is greater than 0
$x_{i,j,k,v}$	A control variable for job position that is equal to 1 if job i is processed on machine k at speed v at stage j, and 0 otherwise
$z_{i,i^*,j,k}$	A control variable for job sequence that is equal to 1 if the next job on the machine k at stage j for job i^* is job i, and 0 otherwise
$B_{k,j}$	The start time of the parallel machine k is at stage j
$E_{k,j}$	The shutdown time of parallel machine k at stage j

4. SDABC of FHFGSP

This section presents the proposed SDABC algorithm for solving FHFGSP. The basic framework of the ABC algorithm is first presented, and then the encoding and decoding scheme and the energy saving procedure are described, followed by the details of SDABC, and finally a summary.

4.1. The Framework of ABC

In ABC, during the initialization phase, a set of food source locations are randomly selected by bees and their nectar amount is determined, then these bees enter the colony and share nectar information. Each search cycle consists of three steps. In the first phase, after information sharing, each employed bee searches for information in the vicinity of the food source location and abandons the old food source to choose a new food source if a better one is found. In the second phase, the onlooker bee selects a food source to follow based on the nectar distribution information sent by the employed bee, the better the food source the more likely it is to be followed. If the current food source is not updated for a long time, the employed bee will abandon the current food source and become a scout bee. The scout bee randomly selects a new food source to replace the abandoned food source. The overall framework of the basic ABC framework is shown in Algorithm 1.

Algorithm 1 Framework of the basic ABC framework

Input: population P;
Output: results;
1: Initialize population P;
2: **while** requirements are met **do**
3: Employed bees to explore around food sources;
4: Onlooker bees select good individuals to follow and explore around the food source;
5: **if** $trial_i$ > threshold limit **then**
6: Employed bees transformed into scout bees looking for new food sources;
7: **end if**
8: **end while**
9: **return** results.

In this paper, the linear weighted sum method is used as the decomposition method. For a multi-objective optimization problem with m objectives, a weight vector $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^T$ is added, where i represents the sum of the weight values of the i-th objective. As shown in (17).

$$\begin{aligned} &\min\ F(X) = \sum_{i=1}^{m} \lambda_i f_i(x_i) \\ &s.t.\ x \in \Omega \end{aligned} \tag{17}$$

where $f_i(x_i)$ is the objective value for the i-th objective. Since this paper is a two-objective problem, $\lambda = (\lambda_1, \lambda_2,)^T$ the values of λ_1 are taken in $\{0/H, 1/H, \ldots, i/H, \ldots, H/H\}$, where $H = N - 1$ and $\lambda_2 = 1 - \lambda_1$.

4.2. Coding Scheme

This subsection gives the encoding and decoding scheme of FHFGSP. The objective of FHFGSP is the minimum MS and TEC. To achieve these two optimization objectives, it is necessary to determine the sequence of jobs in each stage, the machine allocation for each stage of the job, and the speed at which each stage of the job is processed on the machine. Due to the characteristics of FHFGSP, each job needs to go through the same processing stages, so we only need to determine the sequence of jobs into the first stage, and the sequence of jobs in other stages can be determined automatically. However, since the processing speed of each job in each stage is independent of the preceding and following stages and the preceding and following jobs, the processing speed of each job in each stage is independent of the preceding and following jobs. Therefore, the speed of each job in each stage should be determined separately.

Therefore, the solution is coded in two parts. The first part is the sequence of jobs into the first stage. The second part is the velocity selection matrix. In this paper, the two parts of the solution are represented as follows:

$$\begin{aligned} \boldsymbol{\pi}_n &= \{\pi_1, \ldots, \pi_i, \ldots \pi_n\} \\ \mathbf{V}_{m\times n} &= \begin{bmatrix} v_{1,1}, \cdots v_{1,i}, \cdots v_{1,n} \\ \vdots \ \vdots \ \vdots \\ v_{j,1}, \cdots v_{j,i}, \cdots v_{j,n} \\ \vdots \ \vdots \ \vdots \\ v_{m,1}, \cdots v_{m,i}, \cdots v_{m,n} \end{bmatrix} \end{aligned} \tag{18}$$

where $\boldsymbol{\pi}_n$ denotes the n-dimensional job sequence vector, π_i is the sequence number of the i-th job entering the machine, $\mathbf{V}_{m\times n}$ represents the speed matrix of the jobs, and $v_{i,j}$ is the machine processing speed level of the i-th job at stage j.

The second part represents the solution to the three-stage scheduling problem for three jobs as <$\boldsymbol{\pi}_3$, $\mathbf{V}_{3\times3}$>. As shown below, the solution to the three-stage scheduling problem for three jobs is denoted as <$\boldsymbol{\pi}_3$, $\mathbf{V}_{3\times3}$>. $\boldsymbol{\pi}_3$ indicates that the order in which jobs enter the first stage of scheduling is job1, job3, and job2. $\mathbf{V}_{3\times3}$ indicates that in the three stages, job1 is processed at levels 1, 2, and 3, while job2 is processed at levels 2, 1, and 1, and job3 is processed at levels 2, 1, and 3, respectively.

$$\boldsymbol{\pi}_3 = \{1,3,2\}\ \mathbf{V}_{3\times3} = \begin{pmatrix} 1 & 2 & 2 \\ 2 & 1 & 1 \\ 3 & 1 & 3 \end{pmatrix} \tag{19}$$

After the coding scheme is determined, it needs to be decoded into an actual scheduling scheme to make sense. The detailed decoding scheme is as follows. In the first stage, machines are available at the moment 0. According to the order of jobs in $\boldsymbol{\pi}_n$, the jobs are placed on the machine that can be executed earliest and the jobs are processed according to the corresponding speed in the speed matrix, after which the available time of the machine is updated before processing the next job. The following steps are performed for each job in turn in the other stages:

Step 1: Process the job according to its completion time in the previous stage, according to the first-come, first-served principle, i.e., the one that was completed earlier in the previous stage and arrives at this stage first is processed first.

Step 2: Based on the speed in the speed matrix, select the parallel machine that can process the job as early as possible.

Step 3: Update the available time of the machines. Assuming that machine k is available at the moment 0, it takes 3 times to process job i and 1 time to set up, then the available time of the machine is 0 + 3 + 1 = 4 times.

4.3. Initialization and Energy Saving Procedures

After determining the encoding, it is necessary to initialize the populations and external populations. In this paper, the population is initialized in a random way, and for each individual, the job sequence and velocity matrix are generated randomly.

After the population initialization is completed, the dominance relationship between individuals needs to be calculated and the non-dominated solutions are populated with external population.

Although the two optimization objectives of the FHFGSP conflict with each other, it is possible to use a suitable strategy to improve the other objective while controlling one optimization objective constant. To obtain high-quality solutions, individuals use an energy-saving procedure after initialization with the aim of further improving the quality of the population. The basic idea of the energy-saving procedure is to achieve a reduction of PE in TEC by reducing the processing speed of the job while controlling a constant MS.

To achieve constant MS, the energy-saving procedure uses the idea of backtracking. Starting from the last job in the last stage, the processing speed of the job is minimized without affecting the completion time of other jobs. The detailed steps are shown in Algorithm 2, where the symbols that appear are given in Table 1.

Algorithm 2 Energy saving procedure

Input: sequence of assignments in order of completion, $\boldsymbol{\pi}'$;
speed selection matrix, $\mathbf{V}$;
integer related to the number of parallel machines, k;
Output: new speed selection matrix, $\mathbf{V}^*$;
1: i' = the job processed on machine k after i;
2: i^* = the job processed on machine k before i
3: **for** $j = m$ to 1 **do**
4: **for** $l = n$ to 1 **do**
5: $i \leftarrow$ Index of the l-th job in $\boldsymbol{\pi}'$;
6: **for** $v^* = 1$ to $v_{i,j}$ **do**
7: **if** $pt_{i,j,v^*} <= b_{i',j} - st_{i^*,i,j} - st_{i,i',j} - e_{i^*,j}$ **then**
8: $v_{i,j} \leftarrow v^*$;
9: $b_{i,j} = e_{i,j} - pt_{i,j,v^*}$;
10: **break;**
11: **end if**
12: **end for**
13: **end for**
14: **end for**
15: **return** $\mathbf{V}^*$.

4.4. Employed Bees

During the employed bee phase, each individual tries to search around the food source to obtain a better food source. The food source is the solution to the problem.

In order to allow the employed bee to fully explore around the solution, a local search strategy based on ranking is proposed, with the central idea that high-quality solutions are used to guide bad solutions to update themselves.

First, there is a requirement to identify high quality individuals in population. A new way of determining high-quality individuals is proposed. The quality of each individual is related to two factors: the number of dominant solutions and the similarity to the ideal solution. (21) gives a high-quality assessment function for each individual, where n_i denotes the number of solutions in population that are dominated by the current individual i, d^+, and d^- denote the Euclidean distances to the ideal and negative ideal solutions, respectively. (22) gives the formula for the Euclidean distance, where x_i denotes the i-th subproblem of the current solution and x^*_i denotes the i-th subproblem of the ideal solution. Since this paper is about finding a minimum of two objectives, the ideal solution is the lower boundary of the search space and the negative ideal solution is the upper boundary of the search space.

$$value_i = \frac{d_i^-}{d_i^- + d_i^+} + \frac{N - n_i}{N^2} \tag{20}$$

$$d = \sqrt{\left(\sum (x_i - x_i^*)^2\right)} \tag{21}$$

The high-quality individuals then guide the poor individuals to self-renewal when the employed bees search around solutions. The high-quality individuals guided the poor individuals to different degrees, and (23) gives the degree to which each individual i guided the poor individuals. It is worth noting that the high-quality individuals only guide the poorer individuals in their neighborhood. The Euclidean distance of each individual i

in population from other individuals was calculated and the nearest T individuals were selected as neighbors of i.

$$l = \pi * \frac{n_i}{N} \tag{22}$$

In addition, in order to prevent individuals in the population from leading differential updates that affect other individuals that have already been updated and destroy the structure of individuals, individuals in population are sorted in a non-ascending order according to their quality, and individuals that have already been updated do not participate in updates in the same population.

It is also worth noting that five update strategies are used in this paper, depending on the problem to be solved. These strategies are insertion and exchange of working sequences, mutation of velocity matrices, and insertion mutation and cross mutation of working sequences and velocity matrices. The employed bees obtain possible solutions based on these update strategies.

The employed bees search around the solution starting from the first update strategy. If the currently selected update strategy does not yield a solution with high fitness, then the next employed bee searches based on the next update strategy until it finds a high-quality solution. When all five update strategies have been searched, the search starts from the first one again. The flow of the employed bee phase is shown in Algorithm 3, where *Quality*() means calculating the quality of each individual according to (21), *Level*() means determining the degree to which an individual leads the difference solution, *GetNew*() means updating individuals according to the strategy q_i with an initial value of 1 for q_i, and *GetBad*() means obtaining the difference solution that has a high similarity to the current individual and has not been updated.

4.5. Onlooker Bees

In the onlooker bee phase, the onlooker bee will select good food sources for further search based on the information conveyed by the employed bee, with the aim of obtaining high-quality solutions and accelerating the convergence of the algorithm. In this paper, a sorting-based selection strategy is proposed to improve the search efficiency of the onlooker bee and speed up the convergence of the algorithm. First, the individuals in population are ranked according to (21), and those with small values are in the front. The high-quality solutions are placed in front of the bad solutions. Then, the onlooker bee selects an individual in population to follow according to (24), in which (24), i represents the i-th individual to follow and N denotes the population size. Therefore, the individual with the top ranking has a higher probability of being selected.

$$index_i = rand\left(\frac{N+i}{2}\right) \tag{23}$$

After selecting the individual $\boldsymbol{X}_{index}$ according to the selection method proposed in this paper, the onlooker bees randomly select the neighboring individual $\boldsymbol{T}_i$ of the current individual for two-point crossover [41] to generate a new individual. The two-point crossover is divided into two parts: the sequence of operations and the velocity matrix, and the specific operation is as follows: two points in the range are randomly selected, the part between two points in $\boldsymbol{X}_{index}$ is left untouched, and the rest is filled by $\boldsymbol{T}_i$. For the job sequence, the remaining positions in $\boldsymbol{X}_{index}$ are filled by the jobs in $\boldsymbol{T}_i$ that are different from the remaining jobs in $\boldsymbol{X}_{index}$ in turn. For the velocity matrix the remaining positions in $\boldsymbol{X}_{index}$ are filled by the corresponding positions in $\boldsymbol{T}_i$.

Regarding the newly generated individuals, the algorithm will decide whether to replace the original individuals according to the greedy selection algorithm. In particular, in order to prevent the algorithm from falling into local optimum, this paper introduces mutation in the onlooker bee phase, and the probability of mutation of individuals in the population is 1/N. This avoids the algorithm from falling into local optimum to some extent. The whole onlooker bee detailed process is shown in Algorithm 4.

Algorithm 3 Employed bee phase

Input: population P;
Output: new population, $P^{'}$;
$P^{'} = P$;
for $i = 1$ to N **do**
  $s_i = Quality(X_i)$;
end for
$Sort(P^{'}, s)$;
for $i = 1$ to N **do**
  $l_i = Level(X_i)$;
  $X^{'}_i = GetNew(X_i, q_i)$;
  for $z = 0$ to l_i **do**
    if $z = 0$ **then**
    if $X^{'}_i < X_i$ **then**
      $X_i = X^{'}_i$;
    $q_i = 1$;
    else
      $q_i = q_i + 1$;
    end if
    if $q_i > 5$ **then**
      $q_{i=} 1$;
    end if
    else
    X_b=$GetBad(X_i)$;
    if $X^{'}_i < X_b$ **then**
      $X_b = X^{'}_i$;
      break;
    else
      $q_b = q_b + 1$;
    end if
end if
  end for
end for
return P^*.

In Algorithm 4, *Select*() indicates that the onlooker bee selects a food source to follow according to (23), *GetNeighbourhood*() indicates a random selection from the neighbors of the food source, TPX() indicates the two-point crossover, and *Mutation*() represents mutation of an individual **X**, including the job sequence and speed selection matrix.

4.6. Scouting Bees

If a solution is not updated for a long time, the solution will be abandoned and the employed bee will then be transformed into a scout bee, choosing a new solution at random in the solution space. As random search is uncontrollable, this random strategy does not have a positive impact on the algorithm, therefore, this paper uses a neighborhood-based solution swapping strategy to improve the efficiency of the scout bee phase of the algorithm [34]. This is because the solutions of neighboring sub-problems should be similar.

The scout bee searches in the following way: for a solution that has not improved after L cycles, the scout bee first finds a more suitable solution among its neighboring individuals. Then they exchange them with each other. If no better one is found, one individual is randomly chosen to exchange with each other. The basic procedure is described in Algorithm 5, where $L(\mathbf{X}_i)$ denotes the number of cycles $\mathbf{X}_i$ has gone through, T denotes the number of neighboring individuals, $\mathbf{X}_{i,j}$ denotes the j-th neighboring individual of the i-th solution.

Algorithm 4 Onlooker bee phase

Input: population, P;
Output: new population, $P^{'}$;
1: $P^{'} = P$;
2: **for** $j = 1$ to N **do**
3: $s_j = Quality(X_j)$;
4: **end for**
5: Sort($P^{'}$, s_i);
6: **for** $j = 1$ to N **do**
7: X_{index} = Select($P^{'}$)
8: T_i = GetNeighborhood(X_{index});
9: X_{child} = TPX(X_{index},T_i);
10: **if** 0.1 < $Random()$ **then**
11: X_{child} =$Mutation(X_{child})$;
12: **end if**
13: **if** $X_{child} < X_{index}$ **then**
14: X_{index} =X_{child};
15: **end if**
16: **if** $X_{child}) < T_i$ **then**
17: $T_i = X_{child}$;
18: **end if**
19: **end for**
20: **return** $P^{'}$.

Algorithm 5 Scout bee phase

Input: population, P;
Output: new population, $P^{'}$;
1: **for** $i = 1$ to N **do**
2: **if** L(X_i) > L **then**
3: **for** $j = 1$ to T **do**
4: **if** $X_{i,j} < X_i$ **then**
5: $X_{i,j} \leftrightarrow X_i$;
6: **break;**
7: **end if**
8: **end for**
9: **if** $j > T$ **then**
10: r = Rand(1, T);
11: $X_{i,r} \leftrightarrow X_i$;
12: **end if**
13: **end if**
14: **end for**
15: **return** $P^{'}$.

4.7. The Whole Process of the Algorithm

This section outlines the entire algorithmic process of SDABC. It can be roughly divided into four steps.

Step 1: The population is initialized randomly and is energy-efficient to improve the quality of the solution.

Step 2: Sort the population in the manner described in Section 4.4 and perform the algorithmic operations described in Sections 4.5–4.7 in sequence, while updating the domain relationships of individuals in population and the external populations after each subsection is completed.

Step 3: Repeat Step 2 until the end conditions are met.

Step 4: Perform another energy saving procedure on the external population.

The algorithm flow of SDABC can be shown in Figure 1.

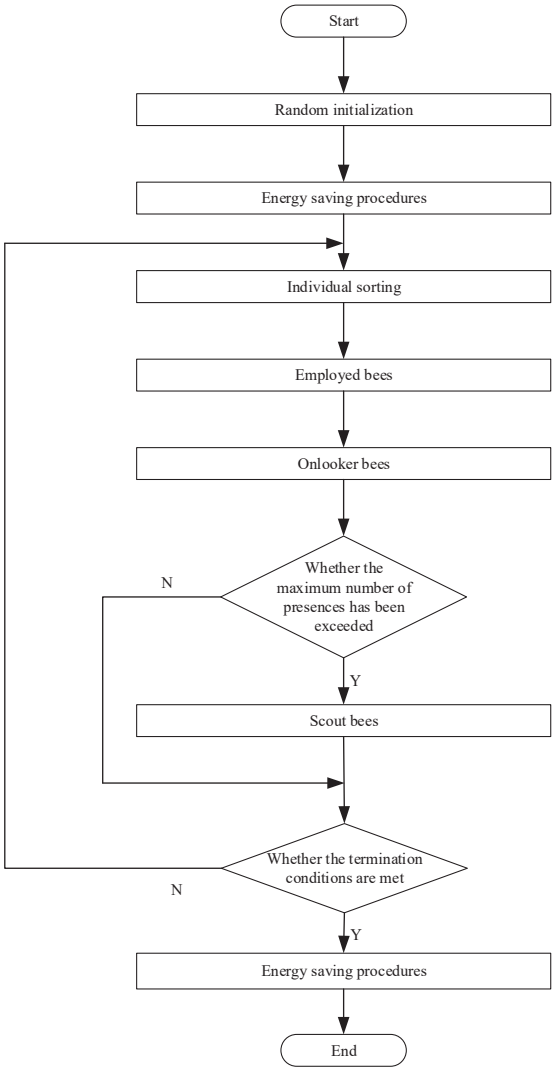


Figure 1. The overall flow of the SDABC.

5. Experiment

In this section, the proposed SDABC algorithm and strategy will be evaluated through experiments. Firstly, the parameter settings of FHFGSP and the performance indicators of the evaluation algorithm are introduced. Then the proposed strategy is compared with other common strategies in experiments. Finally, SDABC is compared with other algorithms in experiments.

The algorithm proposed in this paper is coded in C++ and performed in Codeblocks 16.01. All experiments were run on a PC with an Intel(R) Core (TM) i3-8100U CPU, 3.60 GHz, and 8 GB RAM. Maximum CPU usage time t = 100 was used as a stopping criterion.

5.1. Test Data

In order to fully evaluate the performance of the algorithm from different levels, the performance of SDABC needs to be tested by selecting different problem instances. The

parameters controlling the problem instances are n, m, and st. In this paper, to extensively test the ability of SDABC to solve HFSP of different sizes, five different levels of n, four different levels of m and four different levels of st were designed [34]. This results in 80 problem combinations of different levels. Once the n, m, and st of the problem instances have been determined, it is also necessary to set them separately for the job and the workshop environment. For job$_i$, the processing speed v and the standard processing time p need to be set, and for the shop environment, the number of parallel machines per stage k needs to be set, by means of the previous problem description. It is also necessary to set the energy consumption per unit time of the machines in the processing phase, the setup phase and the idle phase. To avoid chance in the algorithm results, five instances were generated for each problem combination. In summary, the factors and their levels of FHFGSP in generating test data are summarized in Table 2.

Table 2. Summary of test data.

Factors	Levels	Number of Levels
n	20, 40, 60, 80, 100	5
m	3, 5, 8, 10	4
st	U[1, 25], U[1, 49], U[1, 99], U[1, 124]	4
k	U[1, 5]	1
v	U[1, 5]	1
p	U[1, 99]	1
sp	2	1
ip	1	1

5.2. Performance Metrics

Three popular metrics for evaluating multi-objective optimization problems (MOPs) [34,42,43], namely the number of non-dominant solutions, set coverage, and inverse generation distance, were adapted to evaluate the performance of SDABC. The mean and standard deviation of each metric at each level were obtained from 400 instance problems of the FHFGSP over 30 independent iterations.

(1) Number of non-dominated solutions (N-metric). This metric is the number of non-dominant solutions produced by the algorithm, with higher values indicating better performance the closer the PF is.

(2) Inverse Generational Distance (IGD-metric). This metric evaluates the convergence and distribution performance of the algorithm.

$$IGD(\mathbf{A}, \mathrm{PF}^*) = \frac{\sum_{v \in \mathrm{PF}^*} d(v, \mathbf{A})}{\left|\mathrm{PF}^*\right|} \tag{24}$$

where $d(v, \mathbf{A})$ is the minimum Euclidean distance between v and the point in $\mathbf{A}$. The smaller the value, the better the comprehensive performance of the algorithm including convergence and distribution performance. Since the real PF* cannot be solved, all non-dominated solutions obtained jointly by the algorithms of each comparison are used as PF* in this paper.

(3) Set coverage (C-metric). This metric measures the dominance relationship between the two solution sets $\mathbf{A}$ and $\mathbf{B}$.

$$C(\mathbf{A}, \mathbf{B}) = \frac{|\{\mu \in \mathbf{B} | \exists v \in \mathbf{A} : v \prec \mu\}|}{|\mathbf{B}|} \tag{25}$$

where $C(\mathbf{A}, \mathbf{B})$ represents the percentage of ideal solutions in $\mathbf{B}$ that are identical or dominant to those in $\mathbf{A}$. The higher the value, the higher the performance.

In order to eliminate the effect of different metrics, a very simple max-min method [44] is used in this paper to normalize the obtained MS and TEC as follows.

$$f_i = \frac{f_i - \min(f_i)}{\max(f_i) - \min(f_i)} \tag{26}$$

5.3. Effect of Search Strategy

To evaluate the performance of the search strategy, SDABC with a search strategy was compared with DABC without a search strategy. All content factors of the algorithm were the same except for the difference in the employed bee phase search strategy. The evaluation results of the three metrics for the two strategies are shown in Tables A1–A3 (Tables in Appendix A), where the better values are shown in bold and the last row is the average of the 20 problem dimensions.

For the *N*-metric, it can be seen from Table A1 that SDABC has a higher average (AVG) for 85% of the questions and a lower standard deviation (SD) for 75% of the questions. In summary: no problems were found in DABC where both the AVG and SD were better than in SDABC, so SDABC led to better results. For the FHFGSP, for which it is difficult to find the exact solution, a higher *N*-metric can plot the PF more accurately and also help managers to get more options. Therefore, SDABC is more advantageous in this respect.

This is because in the search phase, the employed bee is able to obtain more non-dominated solutions by searching around the individual to different degrees depending on the number of dominant solutions and the similarity of the ideal solutions.

For the *C*-metric, it can be seen from Table A2 that, with the exception for 20×5 and 60×3, SDABC resulted in a better AVG on 90% of the questions and obtained a lower SD on 79% of the questions. Overall SDABC achieved a lower AVG and SD than DABC. To better show the difference between the *C*-metric obtained by SDABC and DABC, a boxplot of the two is plotted in Figure 2, and it can be seen that SDABC is able to obtain more concentrated and dense values and the median was significantly higher for SDABC than DABC. For the values of C (SDABC, DABC) away from the whole, which is the *C*-metric obtained for question 100×5, a comparison of Table A2 shows that lower values were obtained with DABC for the same question.

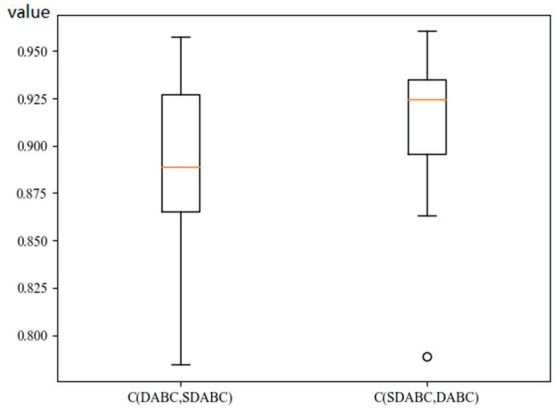


Figure 2. *C*-metric boxplot for DABC and SDABC.

This indicates that the quality of the solutions obtained by SDABC is higher than that of DABC. This is because in the employed bee phase, the employed bee searches around the individual to different degrees based on the similarity between the current solution and the ideal solution, and by being guided by the ideal solution, the employed bee is able to obtain a high-quality solution.

For the *IGD*-metric, it can be seen from Table A3 that SDABC obtains a lower mean value than DABC, except for 20 × 5, 60 × 8, and 80 × 10. For 75% of the questions, SDABC obtained a lower SD. Taken together, SDABC obtained a lower AVG and SD. In order to show the difference more graphically, a boxplot of the two is plotted in Figure 3. It can be seen that SDABC is able to obtain a much more concentrated lower IGD and a much lower median value than DABC.

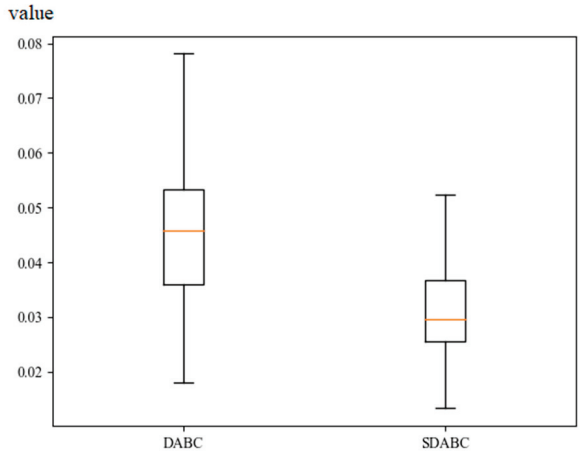


Figure 3. *IGD*-metric boxplot for DABC and SDABC.

This indicates that the solutions obtained by SDABC are better than DABC in terms of diversity and convergence. This is due to the design of five different directions in the search phase, which improves the diversity of solutions obtained. The employed bee improved the convergence by searching around individuals based on the number of dominant solutions and the similarity of ideal solutions.

5.4. Effect of Selection Strategy

Three different selection strategies were compared, in order to evaluate the performance of the newly proposed selection strategy. The three strategies are as follows: the selection strategy proposed in this paper (denoted by ABC_snm), the selection strategy in which individuals in population are selected according to similarity with mutation (denoted by ABC_sm), and the selection strategy in which individuals in population are selected according to similarity without mutation (denoted by ABC_s). The evaluation results of the three metrics for the three strategies are shown in Tables A4–A6, where the better values are shown in bold and the last row is the average of the 20 problem dimensions.

For the *N*-metric, it can be seen from Table A4 that ABC_snm obtained significantly better mean values than ABC_s. Compared to ABC_sm, ABC_snm achieved better results in 80% of the questions. For SD, ABC_s obtained a lower SD value due to the fact that the size of SD is positively related to AVG, and ABC_s has a significantly smaller AVG value, so the resulting SD is also smaller. However, on balance ABC_snm was able to obtain more non-dominated solutions, giving the manager more options to choose from.

This indicates that it is more advantageous to select individuals in the onlooker bee phase based on the number of solutions dominated by them and their similarity to the ideal solution than to select only on the basis of similarity. A comparison of the three can reveal that ABC_snm was able to obtain a greater number of non-dominated solutions.

For *C*-metric, it can be seen from Table A5 that ABC_snm obtains significantly better AVG and SD than ABC_s and ABC_sm. In each problem, ABC_snm achieves better results. Of course, overall, ABC_snm also obtains better AVG and SD than the other two strategies. Figure 4 plots the boxplots of the *C*-metric obtained by the three strategies, and it can be

seen that ABC_snm is able to obtain more concentrated values and obtains a much higher median than the other two strategies, and the minimum value obtained for ABC_snm is also higher than the maximum values of ABC_s and ABC_sn.

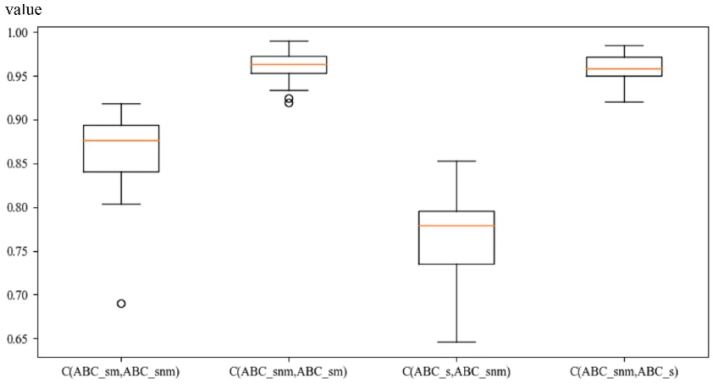


Figure 4. *C*-metric boxplot for ABC_sm, ABC_snm, and ABC_s.

This indicates that in this paper the proposed strategy is able to give obtain high-quality non-dominated solutions. This is due to the fact that the adopted selection strategy can speed up the convergence of the algorithm and the adopted mutation strategy can prevent the algorithm from falling into local optimum.

For *IGD*-metric, it can be seen from Table A6 that in each problem, ABC_snm obtained significantly lower AVG than ABC_s and ABC_sm. In total, 85% of the problems in ABC_snm had smaller SDs than the other two algorithms. As a whole, both the AVG and SD of ABC_snm are smaller than the other two strategies. Figure 5 plots the boxplot of the *IGD*-metric obtained by the three algorithms, and it can be seen that ABC_snm is able to obtain more concentrated values, and the median obtained is much lower than the other two strategies.

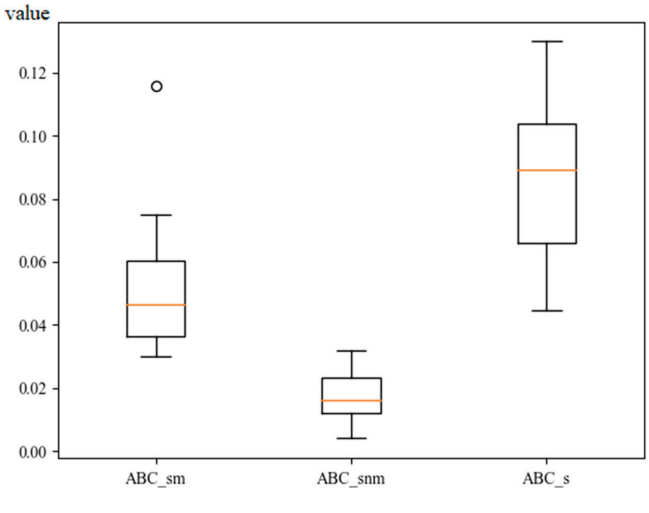


Figure 5. *IGD*-metric boxplot for ABC_sm, ABC_snm, and ABC_s.

This indicates that in this paper the proposed strategy has better diversity and convergence. This is because there is some probability that some low-quality individuals are also selected, which improves the diversity of the algorithm to some extent, and the proposed mutation strategy also has some contribution to the diversity. In addition, the adopted selection strategy can speed up the convergence of the algorithm.

5.5. Evaluation of SDABC

In this subsection, SDABC is compared with IMDABC, MDABC, and NSGAII. All algorithms use the same CPU time as a stopping criterion and all use the same parameter settings, and the results are shown in Tables 8–13 and A7, respectively. The last row of the table represents the average of the 20 problems. The best parts are marked in bold.

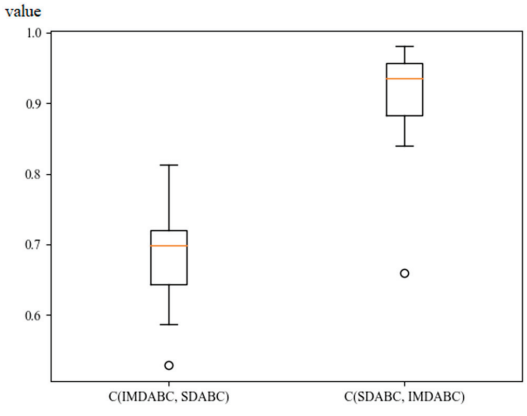
Table A7 shows the *N*-metrics obtained by the four algorithms, and it can be seen that the average values obtained in SDABC are higher than the three remaining algorithms. The values obtained by SDABC are significantly higher than IMDABC and NSGAII in each problem. In addition, although some values of MDABC are higher than SDABC, the difference is not significant, and SDABC achieves higher values in 70% of the problems. To sum up, SDABC is able to obtain more non-dominated solutions compared to other algorithms.

This is because the search strategy proposed by the SDABC in the employed bee phase proposed in this paper is able to search in both depth and breadth directions, enhancing the diversity of individuals and contributing to obtaining a greater number of solutions.

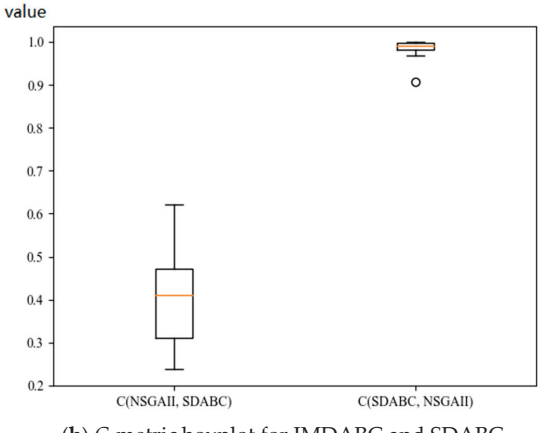
Tables 8–10 show the *C*-metric obtained by the four algorithms, and it can be seen that the AVG and SD obtained by SDABC are significantly higher than IMDABC, MDABC and NSGAII in each of the problems except for the 100×10 problem in Table 10 where the SD is slightly higher. Figure 6 shows a boxplot of the *C*-metric obtained by SDABC versus the other three algorithms. The outliers in (a) are the *C*-metric obtained for problem 100×5. In Table 8, both C-metrics for 100×5 are lower than the overall value, but SDABC's is better than IMDABC's. In (b) it can be seen that SDABC is significantly higher than NSGAII overall. Two independent values of C (SDABC, MDABC) in (c) are for problems 100×3 and 100×5. While these two values deviate from the overall, SDABC has a higher quality AVG and SD for the same problem dimension. Additionally, the median of SDABC is significantly higher than the other three algorithms. Therefore, SDABC obtains solutions of significantly higher quality than IMDABC, MDABC and NSGAII.

This is because SDABC follows the individual in the population in both the employed and onlooker bee phases. The evolution of SDABC continued in accordance with the dominance of individuals in population and the similarity to the ideal solution. At the same time, it is possible to find high-quality solutions faster.

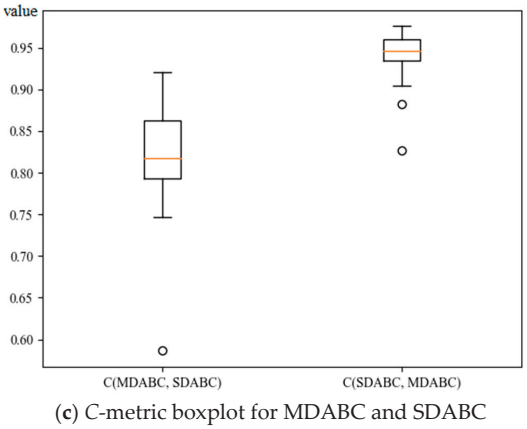
Tables 11–13 show the *IGD*-metric obtained by the four algorithms, and it can be seen that in each problem SDABC obtains significantly lower AVG and SD than IMDABC, MDABC, and NSGAII. Figure 7 plots the boxplot of the *IGD*-metrics obtained by the four algorithms. Figure 7a shows that the overall and median SDABC is much lower than IMDABC. Figure 7b demonstrates that SDABC has a better concentration than MDABC. Figure 7c indicates that SDABC has a better overall and median quality than NSGAII. With Figure 7, we can see that the SDABC distribution is more concentrated under the condition of obtaining a lower *IGD*.



(**a**) C-metric boxplot for IMABC and SDABC

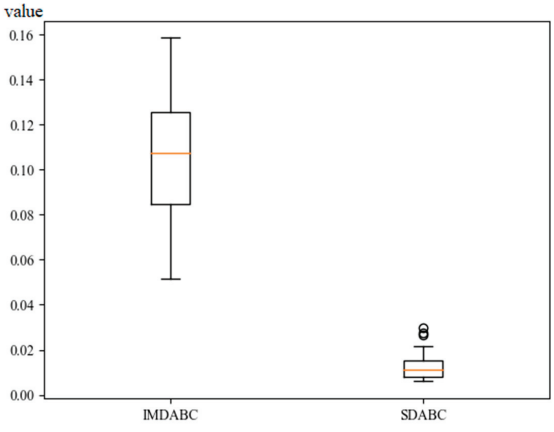


(**b**) C-metric boxplot for IMDABC and SDABC

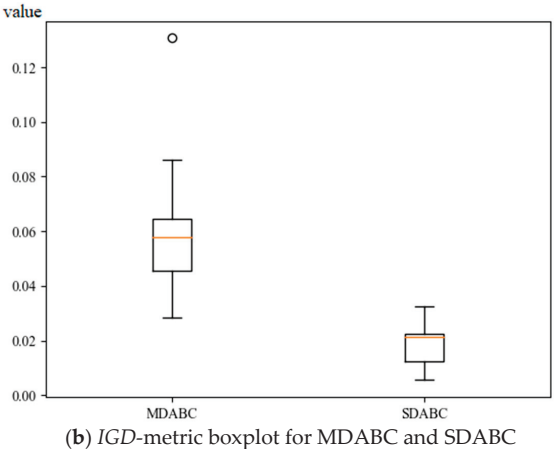


(**c**) C-metric boxplot for MDABC and SDABC

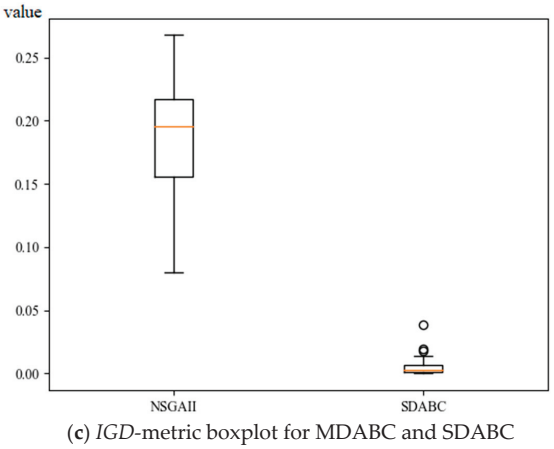
Figure 6. C-metric boxplot for NSGAII, IMDABC, MDABC, and SDABC.



(**a**) *IGD*-metric boxplot for MDABC and SDABC



(**b**) *IGD*-metric boxplot for MDABC and SDABC



(**c**) *IGD*-metric boxplot for MDABC and SDABC

Figure 7. *IGD*-metric boxplot for NSGAII, IMDABC, MDABC, and SDABC.

This indicates that SDABC performs better than IMDABC, MDABC, and NSGAII in terms of convergence and diversity. This is because SDABC takes five different search directions in the employed bee phase and also explores them to different degrees depending on the ranking, both of which can become a more adequate search for individuals in the solution space and increase the diversity of the population. In addition, in the onlooker bee phase, every individual in population has the potential to be tracked. It also contributes to the diversity of the algorithm due to the introduction of the variation strategy. In terms of convergence, both the employed bee and the onlooker bee phases operate based on ranking, which speeds up the convergence of the population based on the similarity and dominance with the ideal solution.

6. Conclusions

In this paper, we studied the FHFGSP with fuzzy processing time that minimizes makespan and total energy consumption. To solve FHFGSP, a discrete artificial bee colony algorithm based on similarity and non-dominated solution ordering was proposed. After extensive numerical experiments, it can be demonstrated that the proposed strategy and algorithm outperforms other algorithms in terms of performance.

In the employed bee phase, individuals fully explore around the dominant solution; in the onlooker bee phase, individuals at the front of the sequence have a greater chance of being followed; in addition, a mutation strategy was proposed to prevent the population from falling into a local optimum. The algorithm produced solutions of high-quality in terms of quantity, quality, convergence, and distribution.

In future, our aim is to study more flexible HFGSPs, such as the proficiency of shop workers, and to consider other green metrics, such as noise and carbon emissions. We will verify the effectiveness of the algorithm by comparing it with more optimization algorithms based on mimicking animal behavior, which will have a positive impact on the role of such algorithms in relation to the green shop scheduling problem. In addition, as smart manufacturing continues to evolve and people start to use information physical systems and industrial Internet of Things to obtain data in real time during manufacturing processes, it is also interesting to study how to process real-time state data for decision making and optimization of green shop scheduling.

Author Contributions: Conceptualization, M.L. and G.-G.W.; methodology, M.L.; software, M.L.; validation, M.L., G.-G.W. and H.Y.; formal analysis, M.L.; investigation, G.-G.W.; resources, H.Y.; data curation, H.Y.; writing—original draft preparation, M.L.; writing—review and editing, M.L., G.-G.W. and H.Y.; visualization, M.L., G.-G.W.; supervision, G.-G.W.; project administration, H.Y. All authors have read and agreed to the published version of the manuscript.

Funding: National Natural Science Foundation of China (No. U19A2061), National Key R&D Program of China (No. 2019YFC1710700), Science and Technology Development Project of Jilin Province (No. 20190301024NY and No. 20200301047RQ).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. *N*-metric for search strategy.

Problem	DABC		SDABC	
	AVG	SD	AVG	SD
20 × 3	86	62	**92**	**59**
20 × 5	65	**36**	**67**	40
20 × 8	45	16	**49**	**16**
20 × 10	40	16	**44**	**15**
40 × 3	106	72	**110**	**71**
40 × 5	54	**30**	**59**	32
40 × 8	34	16	**34**	**15**
40 × 10	**25**	11	23	**9**
60 × 3	79	58	**76**	**52**
60 × 5	50	**21**	**54**	25
60 × 8	32	12	**33**	**11**
60 × 10	26	10	**28**	**9**
80 × 3	61	**53**	**65**	55
80 × 5	32	**20**	**42**	29
80 × 8	**30**	11	28	**10**
80 × 10	22	9	**22**	**8**
100 × 3	58	**46**	**63**	53
100 × 5	**38**	25	36	**24**
100 × 8	25	13	**25**	**12**
100 × 10	22	**9**	**23**	10
Mean	47	**27**	**49**	28

Table A2. *C*-metric for search strategy.

Problem	C(DABC,SDABC)		C(SDABC,DABC)	
	AVG	SD	AVG	SD
20 × 3	0.93880	0.04258	**0.93900**	0.06198
20 × 5	**0.95766**	**0.04046**	0.92896	0.04124
20 × 8	0.88912	0.07329	**0.96045**	**0.03546**
20 × 10	0.88997	0.08919	**0.90871**	**0.07201**
40 × 3	0.93179	**0.05220**	**0.94715**	0.05798
40 × 5	0.88794	0.06988	**0.93452**	**0.05402**
40 × 8	0.86480	0.10236	**0.90333**	**0.07605**
40 × 10	0.86490	0.10413	**0.88452**	**0.06131**
60 × 3	**0.92796**	**0.05899**	0.90222	0.07063
60 × 5	0.86644	0.14676	**0.89354**	**0.10818**
60 × 8	0.92089	0.06839	**0.92381**	**0.05100**
60 × 10	0.90767	0.05409	**0.93009**	**0.04827**
80 × 3	0.86559	0.14008	**0.89635**	**0.07251**
80 × 5	0.79823	0.22431	**0.86336**	**0.15612**
80 × 8	0.87951	0.09336	**0.93812**	**0.06277**
80 × 10	0.92683	0.07375	**0.93418**	**0.04575**
100 × 3	0.84100	0.16037	**0.87990**	**0.13044**
100 × 5	0.78436	**0.18936**	**0.78886**	0.19950
100 × 8	0.88830	0.08372	**0.92509**	**0.07065**
100 × 10	0.92985	**0.05288**	**0.93574**	0.06011
Mean	0.88808	0.09601	**0.91090**	**0.07680**

Table A3. *IGD*-metric for search strategy.

Problem	DABC		SDABC	
	AVG	SD	AVG	SD
20 × 3	0.02663	**0.02198**	**0.01794**	0.03307
20 × 5	**0.01793**	0.02496	0.02749	**0.01907**
20 × 8	0.03927	0.02212	**0.01332**	**0.01327**
20 × 10	0.03363	0.02926	**0.02788**	**0.02733**
40 × 3	0.02985	0.03042	**0.01365**	**0.01382**
40 × 5	0.05140	0.04928	**0.02094**	**0.02754**
40 × 8	0.05918	0.05344	**0.02634**	**0.01598**
40 × 10	0.05839	0.04808	**0.03257**	**0.02078**
60 × 3	0.03715	0.02399	**0.02437**	**0.02303**
60 × 5	0.05341	0.04582	**0.03521**	**0.03348**
60 × 8	**0.02940**	0.03220	0.03209	**0.02915**
60 × 10	0.04229	0.04451	**0.02626**	**0.02061**
80 × 3	0.04921	0.04863	**0.02606**	**0.02082**
80 × 5	0.07811	0.07623	**0.04645**	**0.05436**
80 × 8	0.04589	**0.03531**	**0.03138**	0.04547
80 × 10	**0.03663**	**0.04203**	0.04585	0.05587
100 × 3	0.05310	**0.06431**	**0.05235**	0.06774
100 × 5	0.06235	0.03463	**0.04102**	**0.03457**
100 × 8	0.05340	0.05929	**0.03441**	**0.03360**
100 × 10	0.04582	**0.04686**	**0.04199**	0.06017
Mean	0.04515	0.04167	**0.03088**	**0.03249**

Table A4. *N*-metric for selection strategy.

Problem	ABC_sm		ABC_snm		ABC_s	
	AVG	SD	AVG	SD	AVG	SD
20 × 3	81	55	**92**	59	47	**33**
20 × 5	61	34	**67**	40	50	**31**
20 × 8	40	**13**	**49**	16	42	18
20 × 10	36	**14**	**44**	15	37	17
40 × 3	102	64	**110**	71	51	**35**
40 × 5	55	29	**59**	32	42	**26**
40 × 8	**34**	**14**	**34**	15	32	19
40 × 10	**27**	12	23	**9**	26	12
60 × 3	75	53	**76**	52	35	**28**
60 × 5	53	24	**54**	25	31	**19**
60 × 8	32	10	**33**	11	26	**7**
60 × 10	27	10	**28**	**9**	21	10
80 × 3	57	46	**65**	55	23	**21**
80 × 5	29	18	**42**	29	18	**12**
80 × 8	**30**	11	28	10	20	**7**
80 × 10	**22**	8	**22**	8	16	**6**
100 × 3	60	48	**63**	53	19	**12**
100 × 5	**39**	27	36	24	19	**12**
100 × 8	25	13	**26**	12	13	**6**
100 × 10	**26**	11	23	10	13	**4**
Mean	46	26	**49**	28	29	**17**

Table A5. *C*-metric for selection strategy.

Problem	C(ABC_sm,ABC_snm)		C(ABC_snm,ABC_sm)		C(ABC_s,ABC_snm)		C(ABC_snm,ABC_s)	
	AVG	SD	AVG	SD	AVG	SD	AVG	SD
20 × 3	0.85934	0.08758	**0.98144**	**0.01982**	0.73993	0.17995	**0.97764**	**0.02051**
20 × 5	0.89810	0.05966	**0.96951**	**0.03490**	0.78039	0.16583	**0.98023**	**0.02181**
20 × 8	0.86900	0.11023	**0.96419**	**0.03952**	0.82839	0.10178	**0.97108**	**0.03337**
20 × 10	0.82771	0.13341	**0.97760**	**0.02258**	0.78766	0.12764	**0.98465**	**0.01848**
40 × 3	0.91836	0.04773	**0.97068**	**0.02106**	0.73526	0.16708	**0.97883**	**0.02344**
40 × 5	0.89508	0.08619	**0.95619**	**0.04185**	0.79438	0.16022	**0.97169**	**0.03042**
40 × 8	0.89812	0.06417	**0.96804**	**0.02755**	0.85335	0.12335	**0.96984**	**0.03199**
40 × 10	0.87944	0.06824	**0.96875**	**0.02753**	0.83444	0.11672	**0.95150**	**0.05400**
60 × 3	0.90653	0.06056	**0.95775**	**0.04265**	0.73289	0.17950	**0.95740**	**0.10376**
60 × 5	0.87867	0.09270	**0.95395**	**0.05534**	0.71341	0.16273	**0.94364**	**0.06230**
60 × 8	0.82253	0.09272	**0.98044**	**0.03144**	0.78647	0.13140	**0.95530**	**0.04929**
60 × 10	0.89406	0.06943	**0.95311**	**0.04333**	0.82031	0.14776	**0.95453**	**0.06291**
80 × 3	0.84510	0.10849	**0.97731**	**0.02936**	0.77974	0.21686	**0.96227**	**0.05675**
80 × 5	0.68997	0.20446	**0.98967**	**0.02097**	0.73794	0.21694	**0.95857**	**0.05943**
80 × 8	0.80327	0.13293	**0.96337**	**0.05757**	0.77936	0.10541	**0.94491**	**0.06676**
80 × 10	0.88756	0.08520	**0.93426**	**0.09061**	0.79879	0.15662	**0.96216**	**0.04526**
100 × 3	0.88165	0.12186	**0.95392**	**0.03888**	0.77351	0.19366	**0.95870**	**0.04537**
100 × 5	0.87378	0.10167	**0.94669**	**0.05262**	0.68042	0.21482	**0.92094**	**0.08275**
100 × 8	0.81979	0.11728	**0.92500**	**0.08941**	0.64569	0.19238	**0.94140**	**0.09557**
100 × 10	0.85256	0.09924	**0.91935**	**0.14346**	0.72247	0.17505	**0.93433**	**0.11018**
Mean	0.86003	0.09719	**0.96056**	**0.04652**	0.76624	0.16179	**0.95898**	**0.05372**

Table A6. *IGD*-metric for selection strategy.

Problem	ABC_sm		ABC_snm		ABC_s	
	AVG	SD	AVG	SD	AVG	SD
20 × 3	0.04042	0.03405	**0.00845**	**0.01556**	0.08057	0.03846
20 × 5	0.03659	0.02632	**0.01319**	**0.01536**	0.06153	0.03196
20 × 8	0.03577	0.02567	**0.01321**	**0.01561**	0.04449	0.02785
20 × 10	0.04829	0.03799	**0.00998**	**0.01414**	0.05901	0.03691
40 × 3	0.03020	0.02552	**0.01083**	**0.01343**	0.08398	0.03540
40 × 5	0.03330	**0.03128**	**0.02825**	0.04225	0.06741	0.05066
40 × 8	0.04468	0.03790	**0.02478**	**0.03358**	0.04649	0.02746
40 × 10	0.03573	0.02457	**0.01409**	**0.01484**	0.04507	0.02870
60 × 3	0.02988	0.02740	**0.01928**	**0.02141**	0.10136	0.04475
60 × 5	0.04250	**0.03227**	**0.02603**	0.05360	0.09486	0.06072
60 × 8	0.07492	0.05767	**0.01267**	**0.02564**	0.07959	0.04397
60 × 10	0.04170	0.03099	**0.02281**	**0.02704**	0.06926	0.04548
80 × 3	0.06017	0.05241	**0.01338**	**0.02040**	0.09440	0.07314
80 × 5	0.11588	0.06826	**0.00406**	**0.00849**	0.11549	0.06856
80 × 8	0.07187	0.05380	**0.00723**	**0.00979**	0.09838	0.09451
80 × 10	0.05573	0.05833	**0.02134**	**0.02361**	0.10140	0.09607
100 × 3	0.06039	0.06521	**0.01807**	**0.01793**	0.11416	0.07280
100 × 5	0.05374	0.05680	**0.01827**	**0.01903**	0.11104	0.08333
100 × 8	0.05678	0.05496	**0.03197**	**0.05249**	0.12985	0.08506
100 × 10	0.06252	**0.04959**	**0.03104**	0.05631	0.12392	0.08962
Mean	0.05155	0.04255	**0.01745**	**0.02502**	0.08611	0.05677

Table A7. *N*-metrics for the four algorithms.

Problem	MDABC		SDABC		IMDABC		NSGAII	
	AVG	SD	AVG	SD	AVG	SD	AVG	SD
20 × 3	81	55	**92**	59	47	33	4	**1**
20 × 5	6	34	**73**	39	50	31	4	**1**
20 × 8	40	13	**49**	16	42	18	4	**1**
20 × 10	36	14	**44**	15	37	17	3	**1**
40 × 3	102	64	**110**	71	51	35	4	**1**
40 × 5	55	29	**59**	32	42	26	4	**1**
40 × 8	34	14	**33**	15	32	19	4	**1**
40 × 10	**27**	12	23	9	26	12	3	**1**
60 × 3	75	53	**76**	52	35	28	4	**1**
60 × 5	53	24	**54**	25	31	19	4	**1**
60 × 8	32	10	**33**	11	26	7	4	**1**
60 × 10	27	10	**28**	9	21	10	4	**1**
80 × 3	57	46	**65**	55	23	21	4	**1**
80 × 5	29	18	**42**	29	18	12	4	**1**
80 × 8	**30**	11	28	10	20	7	4	**1**
80 × 10	**22**	8	21	8	16	6	4	**1**
100 × 3	60	48	**63**	53	19	12	4	**1**
100 × 5	**39**	27	36	24	19	12	4	**1**
100 × 8	**26**	13	21	11	13	6	4	**1**
100 × 10	**26**	11	19	7	13	4	4	**1**
Mean	4	26	**49**	28	29	17		**1**

Table 8. *C*-metric for SDABC and IMDABC.

Problem	C(IMDABC, SDABC)		C(SDABC, IMDABC)	
	AVG	SD	AVG	SD
20 × 3	0.71958	0.17690	**0.97863**	**0.02441**
20 × 5	0.81245	0.17156	**0.96822**	**0.04305**
20 × 8	0.69710	0.16276	**0.98021**	**0.02158**
20 × 10	0.67717	0.19304	**0.95016**	**0.06298**
40 × 3	0.67268	0.19374	**0.96970**	**0.04154**
40 × 5	0.71662	0.20731	**0.96423**	**0.03848**
40 × 8	0.71277	0.18846	**0.88374**	**0.09222**
40 × 10	0.65302	0.23329	**0.87723**	**0.09688**
60 × 3	0.68624	0.21262	**0.93698**	**0.09011**
60 × 5	0.58662	0.22109	**0.86391**	**0.18674**
60 × 8	0.75941	0.15186	**0.94086**	**0.04614**
60 × 10	0.74989	0.22071	**0.91005**	**0.08697**
80 × 3	0.69941	0.26287	**0.90214**	**0.12340**
80 × 5	0.60641	0.27873	**0.85887**	**0.17892**
80 × 8	0.72325	0.19220	**0.93444**	**0.13380**
80 × 10	0.79106	0.13459	**0.95489**	**0.05211**
100 × 3	0.61688	0.26191	**0.83928**	**0.25371**
100 × 5	0.52884	0.27615	**0.66027**	**0.32685**
100 × 8	0.60399	0.22069	**0.93966**	**0.10319**
100 × 10	0.70389	0.14102	**0.93097**	**0.10896**
Mean	0.68586	0.20508	**0.91222**	**0.10560**

Table 9. *C*-metric for SDABC and NSGAII.

Problem	C(NSGAII, SDABC)		C(SDABC, NSGAII)	
	AVG	SD	AVG	SD
20 × 3	0.47604	0.19696	**0.99894**	**0.00364**
20 × 5	0.43420	0.21366	**0.99839**	**0.00606**
20 × 8	0.51249	0.18648	**0.99817**	**0.00693**
20 × 10	0.62232	0.28094	**0.97241**	**0.04692**
40 × 3	0.30179	0.20107	**0.99637**	**0.01581**
40 × 5	0.40568	0.24291	**0.98909**	**0.03957**
40 × 8	0.46779	0.23882	**0.99658**	**0.01490**
40 × 10	0.44047	0.25663	**0.99565**	**0.01653**
60 × 3	0.30520	0.23747	**0.99847**	**0.00666**
60 × 5	0.35629	0.20641	**0.99647**	**0.01527**
60 × 8	0.37387	0.25821	**0.98637**	**0.05803**
60 × 10	0.50566	0.27560	**0.98842**	**0.03011**
80 × 3	0.23740	0.18040	**0.98565**	**0.04320**
80 × 5	0.30696	0.23777	**0.99592**	**0.00717**
80 × 8	0.31161	0.27681	**0.98151**	**0.03140**
80 × 10	0.34208	0.30953	**0.90678**	**0.16100**
100 × 3	0.27811	0.21478	**0.98198**	**0.06273**
100 × 5	0.41473	0.25368	**0.99075**	**0.02246**
100 × 8	0.53949	0.31026	**0.97923**	**0.05386**
100 × 10	0.47040	0.32454	**0.96777**	**0.08233**
Mean	0.40513	0.24514	**0.98525**	**0.03623**

Table 10. *C*-metric for SDABC and MDABC.

Problem	C(MDABC, SDABC)		C(SDABC, MDABC)	
	AVG	SD	AVG	SD
20 × 3	0.85678	0.11097	**0.97497**	**0.02959**
20 × 5	0.92044	0.05882	**0.96118**	**0.04077**
20 × 8	0.83238	0.09010	**0.97593**	**0.03354**
20 × 10	0.82119	0.12392	**0.95335**	**0.05121**
40 × 3	0.90210	0.06388	**0.96763**	**0.03469**
40 × 5	0.86154	0.10690	**0.95649**	**0.04270**
40 × 8	0.81330	0.11659	**0.90450**	**0.09811**
40 × 10	0.77790	0.12432	**0.92043**	**0.05911**
60 × 3	0.90741	0.06411	**0.93831**	**0.05046**
60 × 5	0.81425	0.13803	**0.92594**	**0.07804**
60 × 8	0.79762	0.10058	**0.96047**	**0.04091**
60 × 10	0.86893	0.07453	**0.95030**	**0.04694**
80 × 3	0.78871	0.19319	**0.94319**	**0.05262**
80 × 5	0.58628	0.21435	**0.95776**	**0.10780**
80 × 8	0.74688	0.13289	**0.97491**	**0.03945**
80 × 10	0.86783	0.10606	**0.93900**	**0.10327**
100 × 3	0.79428	0.16286	**0.88190**	**0.11669**
100 × 5	0.75590	0.17291	**0.82682**	**0.15634**
100 × 8	0.79468	0.14633	**0.93958**	**0.10413**
100 × 10	0.84412	**0.10980**	**0.93745**	0.13495
Mean	0.81763	0.12056	**0.93951**	**0.07107**

Table 11. *IGD*-metric for SDABC and IMDABC.

Problem	IMDABC		SDABC	
	AVG	SD	AVG	SD
20 × 3	0.09976	0.03581	**0.00663**	**0.01234**
20 × 5	0.05147	0.03417	**0.00869**	**0.01174**
20 × 8	0.07184	0.03675	**0.01113**	**0.01998**
20 × 10	0.07962	0.03873	**0.01176**	**0.01615**
40 × 3	0.11765	0.04426	**0.00600**	**0.00950**
40 × 5	0.10199	0.05379	**0.00720**	**0.00759**
40 × 8	0.07725	0.05090	**0.02156**	**0.01843**
40 × 10	0.08825	0.05355	**0.02981**	**0.02419**
60 × 3	0.12603	0.06153	**0.00790**	**0.00855**
60 × 5	0.12552	0.05599	**0.01545**	**0.01365**
60 × 8	0.07575	0.04625	**0.01527**	**0.01209**
60 × 10	0.08662	0.06164	**0.02675**	**0.04270**
80 × 3	0.12157	0.10182	**0.00800**	**0.01224**
80 × 5	0.15439	0.09136	**0.01543**	**0.02934**
80 × 8	0.11297	0.08460	**0.00951**	**0.01343**
80 × 10	0.09190	0.06572	**0.01140**	**0.01092**
100 × 3	0.15286	0.08546	**0.01482**	**0.02365**
100 × 5	0.15846	0.09721	**0.02767**	**0.03839**
100 × 8	0.14235	0.09122	**0.01025**	**0.01173**
100 × 10	0.11993	0.09412	**0.00646**	**0.00858**
Mean	0.10781	0.06424	**0.01358**	**0.01726**

Table 12. *IGD*-metric for SDABC and MDABC.

Problem	MDABC		SDABC	
	AVG	SD	AVG	SD
20 × 3	0.04353	0.03879	**0.00829**	**0.01226**
20 × 5	0.02825	0.02566	**0.01355**	**0.01575**
20 × 8	0.05191	0.02136	**0.00551**	**0.00716**
20 × 10	0.05523	0.03113	**0.01197**	**0.01563**
40 × 3	0.03621	0.02506	**0.01228**	**0.01483**
40 × 5	0.04554	0.03478	**0.02105**	**0.02939**
40 × 8	0.06442	0.05616	**0.03238**	**0.04255**
40 × 10	0.06916	0.04299	**0.02161**	**0.02047**
60 × 3	0.04003	0.02879	**0.02216**	**0.02420**
60 × 5	0.04527	0.02730	**0.02194**	**0.02262**
60 × 8	0.06443	0.04546	**0.01398**	**0.02034**
60 × 10	0.04712	0.03247	**0.01629**	**0.01596**
80 × 3	0.07120	0.06431	**0.02195**	**0.03317**
80 × 5	0.13071	0.06473	**0.00966**	**0.01330**
80 × 8	0.08625	0.04796	**0.00926**	**0.01318**
80 × 10	0.05353	0.03556	**0.02344**	**0.02623**
100 × 3	0.06290	0.05063	**0.02225**	**0.02473**
100 × 5	0.06074	0.03328	**0.02962**	**0.02276**
100 × 8	0.06374	0.05757	**0.02619**	**0.04760**
100 × 10	0.06049	0.04981	**0.02627**	**0.04505**
Mean	0.05903	0.04069	**0.01848**	**0.02336**

Table 13. *IGD*-metric for SDABC and NSGAII.

Problem	NSGAII		SDABC	
	AVG	SD	AVG	SD
20 × 3	0.11426	0.05009	**0.00008**	**0.00028**
20 × 5	0.12345	0.03838	**0.00003**	**0.00013**
20 × 8	0.12013	0.04920	**0.00024**	**0.00098**
20 × 10	0.07980	0.04905	**0.01939**	**0.03984**
40 × 3	0.18873	0.09539	**0.00145**	**0.00633**
40 × 5	0.14942	0.08224	**0.00299**	**0.00774**
40 × 8	0.15783	0.08214	**0.00319**	**0.01391**
40 × 10	0.16789	0.09567	**0.00303**	**0.00927**
60 × 3	0.19377	0.08821	**0.00019**	**0.00082**
60 × 5	0.20634	0.10966	**0.00194**	**0.00843**
60 × 8	0.23376	0.13245	**0.00425**	**0.00973**
60 × 10	0.19724	0.10722	**0.00435**	**0.00995**
80 × 3	0.21405	0.10015	**0.00240**	**0.00795**
80 × 5	0.22440	0.11524	**0.00139**	**0.00267**
80 × 8	0.25470	0.16455	**0.01070**	**0.01762**
80 × 10	0.20225	0.16029	**0.03841**	**0.05594**
100 × 3	0.22540	0.12751	**0.00505**	**0.01453**
100 × 5	0.19644	0.10497	**0.00318**	**0.00710**
100 × 8	0.19358	0.16121	**0.01789**	**0.02596**
100 × 10	0.26764	0.19026	**0.01394**	**0.03123**
Mean	0.18555	0.10519	**0.00670**	**0.01352**

References

1. Ruiz, R.; Vázquez-Rodríguez, J.A. The hybrid flow shop scheduling problem. *Eur. J. Oper. Res.* **2010**, *205*, 1–18. [CrossRef]
2. Qin, T.; Du, Y.; Chen, J.H.; Sha, M. Combining mixed integer programming and constraint programming to solve the integrated scheduling problem of container handling operations of a single vessel. *Eur. J. Oper. Res.* **2020**, *285*, 884–901. [CrossRef]
3. Pan, Q.; Wang, L.; Mao, K.; Zhao, J.; Zhang, M. An effective artificial bee colony algorithm for a real-world hybrid flowshop problem in steelmaking process. *IEEE Trans. Autom. Sci. Eng.* **2013**, *10*, 307–322. [CrossRef]
4. Li, J.Q.; Pan, Q.K.; Mao, K. A hybrid fruit fly optimization algorithm for the realistic hybrid flowshop rescheduling problem in steelmaking systems. *IEEE Trans. Autom. Sci. Eng.* **2016**, *13*, 932–949. [CrossRef]
5. Lin, Y.K.; Huang, D.H. Reliability analysis for a hybrid flow shop with due date consideration. *Reliab. Eng. Syst. Saf.* **2020**, *199*, 105905. [CrossRef]
6. Lin, P.; Shen, L.; Zhao, Z.; Huang, G.Q. Graduation manufacturing system: Synchronization with IoT-enabled smart tickets. *J. Intell. Manuf.* **2019**, *30*, 2885–2900. [CrossRef]
7. Ali, D.; Frimpong, S. Artificial intelligence models for predicting the performance of hydro pneumatic suspension struts in large capacity dump trucks. *Int. J. Ind. Ergon.* **2018**, *67*, 283–295. [CrossRef]
8. Worasan, K.; Sethanan, K.; Pitakaso, R.; Moonsri, K.; Nitisiri, K. Hybrid particle swarm optimization and neighborhood strategy search for scheduling machines and equipment and routing of tractors in sugarcane field preparation. *Comput. Electron. Agric.* **2020**, *178*, 105733. [CrossRef]
9. Fortemps, P. Jobshop scheduling with imprecise durations: A fuzzy approach. *IEEE Trans. Fuzzy Syst.* **1997**, *5*, 557–569. [CrossRef]
10. Lei, D.; Gao, L.; Zheng, Y. A Novel Teaching-Learning-Based Optimization Algorithm for Energy-Efficient Scheduling in Hybrid Flow Shop. *IEEE Trans. Eng. Manag.* **2018**, *65*, 330–340. [CrossRef]
11. Zhang, Z.; Tang, R.; Peng, T.; Tao, L.; Jia, S. A method for minimizing the energy consumption of machining system: Integration of process planning and scheduling. *J. Clean. Prod.* **2016**, *137*, 1647–1662. [CrossRef]
12. Wang, L.; Wang, J.; Wu, C. Advances in green shop scheduling and optimization. *Control Decis.* **2018**, *33*, 385–391.
13. Fu, Y.; Zhou, M.; Guo, X.; Qi, L. Scheduling Dual-Objective Stochastic Hybrid Flow Shop With Deteriorating Jobs via Bi-Population Evolutionary Algorithm. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *50*, 5037–5048. [CrossRef]
14. Yankai, W.; Shilong, W.; Dong, L.; Chunfeng, S.; Bo, Y. An improved multi-objective whale optimization algorithm for the hybrid flow shop scheduling problem considering device dynamic reconfiguration processes. *Expert Syst. Appl.* **2021**, *174*, 114793. [CrossRef]
15. Gao, H.; Shi, Y.; Pun, C.; Kwong, S. An improved artificial bee colony algorithm with its application. *IEEE Trans. Ind. Inf.* **2019**, *15*, 1853–1865. [CrossRef]
16. Gao, W.; Liu, S.; Huang, L. A novel artificial bee colony algorithm based on modified search equation and orthogonal learning. *IEEE Trans. Cybern.* **2013**, *43*, 1011–1024. [CrossRef] [PubMed]

17. Karaboga, D.; Akay, B. A comparative study of Artificial Bee Colony algorithm. *Appl. Math. Comput.* **2009**, *214*, 108–132. [CrossRef]
18. Li, J.; Pan, Q. Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm. *Inf. Sci.* **2015**, *316*, 487–502. [CrossRef]
19. Wang, F.; Rao, Y.; Zhang, C.; Tang, Q.; Zhang, L. Estimation of distribution algorithm for energy-efficient scheduling in turning processes. *Sustainability* **2016**, *8*, 762. [CrossRef]
20. Xuan, H.; Zhag, H.; Li, B. An Improved Discrete Artificial Bee Colony Algorithm for Flexible Flowshop Scheduling with Step Deteriorating Jobs and Sequence-Dependent Setup Times. *Math. Probl. Eng.* **2019**, *2019*, 1–13. [CrossRef]
21. Yue, L.; Guan, Z.; Zhang, L.; Ullah, S.; Cui, Y. Multi objective lotsizing and scheduling with material constraints in flexible parallel lines using a Pareto based guided artificial bee colony algorithm. *Comput. Ind. Eng.* **2019**, *128*, 659–680. [CrossRef]
22. Gong, G.; Chiong, R.; Deng, Q.; Gong, X. A hybrid artificial bee colony algorithm for flexible job shop scheduling with worker flexibility. *Int. J. Prod. Res.* **2020**, *58*, 4406–4420. [CrossRef]
23. Zadeh, M.S.; Katebi, Y.; Doniavi, A. A heuristic model for dynamic flexible job shop scheduling problem considering variable processing times. *Int. J. Prod. Res.* **2019**, *57*, 3020–3035. [CrossRef]
24. Lei, D.; Liu, M. An artificial bee colony with division for distributed unrelated parallel machine scheduling with preventive maintenance. *Comput. Ind. Eng.* **2020**, *141*, 106320. [CrossRef]
25. Xie, Z.; Yang, D.; Ma, M.; Yu, X. An Improved Artificial Bee Colony Algorithm for the Flexible Integrated Scheduling Problem Using Networked Devices Collaboration. *Int. J. Coop. Inf. Syst.* **2020**, *29*, 2040003. [CrossRef]
26. Li, J.; Bai, S.; Duan, P.; Sang, H.; Han, Y.; Zheng, Z. An improved artificial bee colony algorithm for addressing distributed flow shop with distance coefficient in a prefabricated system. *Int. J. Prod. Res.* **2019**, *57*, 6922–6942. [CrossRef]
27. Li, J.; Song, M.; Wang, L.; Duan, P.; Han, Y.; Sang, H.; Pan, Q. Hybrid Artificial Bee Colony Algorithm for a Parallel Batching Distributed Flow-Shop Problem With Deteriorating Jobs. *IEEE Trans. Cybern.* **2020**, *50*, 2425–2439. [CrossRef]
28. Gong, D.; Han, Y.; Sun, J. A novel hybrid multi-objective artificial bee colony algorithm for blocking lot-streaming flow shop scheduling problems. *Knowl. -Based Syst.* **2018**, *148*, 115–130. [CrossRef]
29. Pan, Q.; Gao, L.; Wang, L.; Liang, J.; Li, X. Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem. *Expert Syst. Appl.* **2019**, *124*, 309–324. [CrossRef]
30. Li, X.; Tang, H.; Yang, Z.; Wu, R.; Luo, Y. Integrated Optimization Approach of Hybrid Flow-Shop Scheduling Based on Process Set. *IEEE Access* **2020**, *8*, 223782–223796. [CrossRef]
31. Peng, K.; Pan, Q.; Gao, L.; Zhang, B.; Pang, X. An Improved Artificial Bee Colony algorithm for real-world hybrid flowshop rescheduling in Steelmaking-refining-Continuous Casting process. *Comput. Ind. Eng.* **2018**, *122*, 235–250. [CrossRef]
32. Zhong, Y.; Yang, F.; Liu, F. Solving multi-objective fuzzy flexible job shop scheduling problem using MABC algorithm. *J. Intell. Fuzzy Syst.* **2019**, *36*, 1455–1473. [CrossRef]
33. Li, Y.; Huang, W.; Wu, R.; Guo, K. An improved artificial bee colony algorithm for solving multi-objective low-carbon flexible job shop scheduling problem. *Appl. Soft Comput.* **2020**, *95*, 106544. [CrossRef]
34. Zhang, B.; Pan, Q.; Gao, L.; Li, X.; Meng, L.; Peng, K. A multiobjective evolutionary algorithm based on decomposition for hybrid flowshop green scheduling problem. *Comput. Ind. Eng.* **2019**, *136*, 325–344. [CrossRef]
35. Linn, R.; Zhang, W. Hybrid flow shop scheduling: A survey. *Comput. Ind. Eng.* **1999**, *37*, 57–61. [CrossRef]
36. Ribas, I.; Leisten, R.; Framinan, J.M. Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Comput. Oper. Res.* **2010**, *37*, 1439–1454. [CrossRef]
37. Chang, D.Y. Applications of the extent analysis method on fuzzy AHP. *Eur. J. Oper. Res.* **1996**, *95*, 649–655. [CrossRef]
38. Fang, K.; Uhan, N.; Zhao, F.; Sutherland, J.W. A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. *J. Manuf. Syst.* **2011**, *30*, 234–240. [CrossRef]
39. Zadeh, L.A. Fuzzy logic. *Computer* **1988**, *21*, 83–93. [CrossRef]
40. Gao, D.; Wang, G.; Pedrycz, W. Solving fuzzy job-shop scheduling problem using DE algorithm improved by a selection mechanism. *IEEE Trans. Fuzzy Syst.* **2020**, *28*, 3265–3275. [CrossRef]
41. Ozturk, C.; Hancer, E.; Karaboga, D. A novel binary artificial bee colony algorithm based on genetic operators. *Inf. Sci.* **2015**, *297*, 154–170. [CrossRef]
42. Fathollahi-Fard, A.M.; Hajiaghaei-Keshteli, M.; Mirjalili, S. Multi-objective stochastic closed-loop supply chain network design with social considerations. *Appl. Soft Comput.* **2018**, *71*, 505–525. [CrossRef]
43. Riquelme, N.; Lücken, C.V.; Baran, B. Performance metrics in multi-objective optimization. In Proceedings of the 2015 Latin American Computing Conference (CLEI 2015), Arequipa, Peru, 19 October 2015; pp. 1–11.
44. Zhang, Q.; Li, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **2007**, *11*, 712–731. [CrossRef]

MDPI
St. Alban-Anlage 66
4052 Basel
Switzerland
Tel. +41 61 683 77 34
Fax +41 61 302 89 18
www.mdpi.com

Mathematics Editorial Office
E-mail: mathematics@mdpi.com
www.mdpi.com/journal/mathematics

MDPI
St. Alban-Anlage 66
4052 Basel
Switzerland

Tel: +41 61 683 77 34
Fax: +41 61 302 89 18

www.mdpi.com

ISBN 978-3-0365-2395-8