applied sciences

MDPI

Article

Comparative Analysis of Selection Hyper-Heuristics for Real-World Multi-Objective Optimization Problems

Vinicius Renan de Carvalho [1], Ender Özcan [2] and Jaime Simão Sichman [1,*]

[1] Laboratório de Técnicas Inteligentes (LTI), Escola Politécnica (EP), Universidade de São Paulo (USP), São Paulo 05508-970, Brazil; vrcarvalho@usp.br
[2] Computational Optimisation and Learning (COL) Lab, School of Computer Science, University of Nottingham, Nottingham NG8 1BB, UK; ender.ozcan@nottingham.ac.uk
* Correspondence: jaime.sichman@usp.br

Abstract: As exact algorithms are unfeasible to solve real optimization problems, due to their computational complexity, meta-heuristics are usually used to solve them. However, choosing a meta-heuristic to solve a particular optimization problem is a non-trivial task, and often requires a time-consuming trial and error process. Hyper-heuristics, which are heuristics to choose heuristics, have been proposed as a means to both simplify and improve algorithm selection or configuration for optimization problems. This paper novel presents a novel cross-domain evaluation for multi-objective optimization: we investigate how four state-of-the-art online hyper-heuristics with different characteristics perform in order to find solutions for eighteen real-world multi-objective optimization problems. These hyper-heuristics were designed in previous studies and tackle the algorithm selection problem from different perspectives: Election-Based, based on Reinforcement Learning and based on a mathematical function. All studied hyper-heuristics control a set of five Multi-Objective Evolutionary Algorithms (MOEAs) as Low-Level (meta-)Heuristics (LLHs) while finding solutions for the optimization problem. To our knowledge, this work is the first to deal conjointly with the following issues: (i) selection of meta-heuristics instead of simple operators (ii) focus on multi-objective optimization problems, (iii) experiments on real world problems and not just function benchmarks. In our experiments, we computed, for each algorithm execution, Hypervolume and IGD+ and compared the results considering the Kruskal–Wallis statistical test. Furthermore, we ranked all the tested algorithms considering three different Friedman Rankings to summarize the cross-domain analysis. Our results showed that hyper-heuristics have a better cross-domain performance than single meta-heuristics, which makes them excellent candidates for solving new multi-objective optimization problems.

Keywords: artificial intelligence; evolutionary algorithms; multi-objective optimization; meta-heuristics; genetic algorithms; online algorithm selection; hyper-heuristics

Citation: de Carvalho, V.R.; Özcan, E.; Sichman, J.S. Comparative Analysis of Selection Hyper-Heuristics for Real-World Multi-Objective Optimization Problems. *Appl. Sci.* **2021**, *11*, 9153. https://doi.org/10.3390/app11199153

Academic Editor: Juan Francisco De Paz Santana

Received: 18 July 2021
Accepted: 20 September 2021
Published: 1 October 2021

1. Introduction

Choosing a meta-heuristic to solve a particular optimization problem is a non-trivial task. Without detailed prior information as to which particular algorithm to use, it demands the evaluation of several algorithms in order to find out which is more suitable to solve a given problem. However, due to the non-deterministic nature of these algorithms, this process demands to be repeated several times. Hyper-heuristics, which are *heuristics to choose heuristics*, have been proposed as a means to both simplify and improve algorithm selection or configuration for optimization problems [1,2]. The idea is, through automation of the heuristic search, to provide effective and reusable cross-domain search methodologies that are applicable to the problems with different characteristics from various domains without requiring much expert involvement [3].

Hyper-heuristics (HH) employ learning methods, by using some feedback from the search process. Based on the source of this feedback, HHs can be classified as online or offline. A hyper-heuristic employs online learning if learning takes place while the algorithm

is solving an instance of a problem. It is offline if the knowledge is gathered in the form of rules or programs from a set of training instances that hopefully generalize to solving unseen instances [1,2].

Hyper-heuristics can also be classified as selection or generation methodologies [4,5]. Selection hyper-heuristics at the high-level control and mix low-level (meta)heuristics (LLHs), automatically deciding which one(s) to apply to the candidate solution(s) at each decision point of the iterative search process [4]. On the other hand, generation methodologies produce new heuristics or heuristic parts using pre-defined components.

Much of the previous research focuses on online selection hyper-heuristics. The interest in such algorithms has been growing in recent years, but the majority of research in this area has been limited to single-objective optimization [6].

The majority of the research in the HH literature focus on treating operators, such as crossover, mutation, and differential evolution as LLH [2,6]. In [7], Cowling et al. proposed the *Choice Function*, an equation responsible for rank heuristics considering the algorithm performance, in this case, a fitness function for a mono-objective problem, and the computational time.

In [8], Li et al. introduced FFRMAB, a variation of the original Multi-Armed Bandit [9] (MAB) where the *Fitness Rate Ranking* (*FRR*) was proposed as reward assignment. In this hyper-heuristic, a set of Differential Evolution operators was considered as LLH for being chosen. Following the choice of DE operators, Gonçalves et al. [10] proposed a hyper-heuristic based on the choice function to control a set of five Differential Evolution operators. Results showed this hyper-heuristic overcoming the performance of the standard MOEA (using a single operator) when solving ten unconstrained benchmark functions with two and three objectives. In [11], the authors applied a similar approach using several versions of MAB instead of using a Choice Function. In this case, the CEC 2009 benchmark [12] was employed for performance evaluation. In [13], Almeida et al. evaluated three different versions of MAB by applying them to the permutation flow shop problem.

In [14], Guizzo et al. designed a hyper-heuristic to solve the Class Integration Test Order Problem [15], a software engineering problem where nodes from a graph have to be visited, where these nodes are the classes to be tested. This hyper-heuristic was built using a choice function [6] and a Multi-Armed Bandit [9] to select a LLH from a set of nine to operate together with a fixed MOEA, in this case, the NSGA-II algorithm. This set was built by combining different crossover and mutation operators. The evaluation of LLHs was performed based on the dominance relationship among parent solutions and their offspring. In [16], this approach was tested considering SPEA2 as the fixed MOEA. In [17], de Carvalho tacked the same problem by creating a hyper-heuristic based on FRRMAB [8] and considering the same set of LLHs. Among all these versions, the Choice Function applied together with NSGA-II ([14] version) obtained the best results.

Only a few studies focus on online learning hyper-heuristics selecting/mixing multi-objective evolutionary algorithms (MOEAs) for solving multi-objective optimization problems. Among them, Maashi et al. [6] proposed the Choice Function Hyper-Heuristic (HHCF) which is an interesting approach employing different quality indicators to evaluate a set of LLHs. In this approach, each LLH executes for some generations, then the resulting population is evaluated based on m different quality indicators, generating a table $n \times m$, where n is the number of LLHs, and m is the number of quality indicators. Following that, a second table containing the rankings is generated, which is used as input to a choice mathematical function responsible for determining which LLH should execute next. This approach was tested using the Walking Fish Group (WFG) benchmark [18] and on a single real-world problem: the Crashworthiness problem [19]. The authors used the algorithms NSGAII [20], SPEA2 [21], and MOGA [22] as LLHs, and compared their results to Amalgam [23] and to all single MOEAs. The experimental results indicated the success of HHCF outperforming them all. In [24], Li et al. replaced MOGA by IBEA [25], and evaluated the approach on solving another real-world problem: Wind Farm layout optimization [26]. Although this hyper-heuristic yielded good results, the use of a two-level ranking approach was not justified properly and there was no theoretical background to it.

The following studies have employed well-established computational methods within the design of new hyper-heuristics. Li et al. [3] proposed MOHH-RILA, an online selection hyper-heuristic treating the problem as a learning automata, where the automata action is the selection of an LLH from a set of MOEAs. As *reward* and *penalty*, they used Hypervolume and Spread indicators (the last in case of ties). Thus, this study can be classified as a Reinforcement Learning based online selection hyper-heuristic. The authors employed IBEA, SPEA2, and NSGAII as LLHs to find solutions for the WFG and DTLZ [27] benchmarks and variants of the crashworthiness problem. The results showed that this approach outperformed HHCF, making it the state-of-the-art online selection hyper-heuristic for multi-objective optimization.

de Carvalho et al. [28,29] proposed the Multi-Objective Agent-Based Hyper-Heuristic (MOABHH), an online selection hyper-heuristic enabling the best performing LLH to have a greater share of the new solutions/offspring. They designed this HH as a multi-agent system by treating all LLH and quality indicators as agents and employed Social Choice Theory [30] voting methods in order to summarize quality indicators preferences for the participation assignment mechanism. In the first study [28], they employed NSGAII, IBEA, and SPEA2 as LLHs and applied MOABHH to the WFG Suite. GDE3 [31] was additionally included within the LLH set in [32]. In the latter [29], they evaluated how MOABHH performs on real-world problems, including the Crashworthiness, Water [33], Car Side Impact [34], and Machining [35]. In [36], the authors evaluated the proposed approach considering different voting criterion considering these four real-world problems and the multi-objective travel salesperson problem [37]. This proposed HH outperformed each individual MOEA and random choice hyper-heuristic; however, MOABHH was not compared to any of the other state-of-the-art hyper-heuristics.

Hence, the goal of this paper is to perform a thorough investigation of four reportedly top-ranking hyper-heuristics across an extensive set of problems: (i) MOABHH, which achieved promising results in a previous study; (ii) two variants of MOHH-RILA (HHLA and HHRL); and (iii) HHCF.

We used eighteen real-world multi-objective problems in total: four that were already used in [32], ten problems from the *Black Box Optimization Competition* [38], and four other bi-objective problems from FourBarTuss [39], Goliski [40], Quagliarella [41], and Poloni [42]. We also increased the number of LLHs used by all HHs to five: NSGAII, IBEA, SPEA2, GDE3, and mIBEA [43]. As a consequence, the task of selecting a LLH at each decision point was harder for all tested HHs.

This paper focuses on comparing selection online hyper-heuristics specialized in selecting multi-objective evolutionary algorithms. For this reason, hyper-heuristics focused on selecting/generating parts of algorithms such as heuristics (i.e., crossover and mutation) can not be considered. This is also the case of micro-genetic algorithms [44], which also focus on selecting part of the evolutionary algorithms. Other approaches, such as parameter control [45], which focuses on diminishing the effort on setting up parameters by automatically setting them, will also not be considered since they are not directly related to hyper-heuristics.

The rest of the paper is organized as follows: Section 2 describes the studied hyper-heuristics and succinctly describes MOEAs employed as low-level heuristics. Section 3 describes our methodology for the performed experiments. In Section 4, we present and discuss our obtained results. Finally, we present our conclusions and further work in Section 5.

2. Background

2.1. Multi-Objective Optimization

Multi-objective optimization (MOPs) consists of finding solutions which simultaneously consider two or more conflicting objectives to be minimized or maximized. Thus, the search aims to find a set of solutions, each one reflecting a trade-off between the objectives. MOPs are tackled today using Evolutionary Algorithms by engineers, computer scientists, biologists, and operations researchers alike [46]. These algorithms are heuristic techniques that allow a flexible representation of the solutions and do not impose continuity

conditions on the functions to be optimized. Moreover, MOEAs are extensions of EAs for multi-objective problems that usually apply the concepts of Pareto dominance [47].

MOEAs have been applied to solve MOPs from different areas, from logistic problems as the Ridesharing problem [48], Software Engineering problems as architecture optimization [49], machine learning problems such as feature selection [50], and by optimizing antibiotic treatments [51].

Besides EAs, there are other nature-based algorithms that have been successfully applied to solve optimization problems. This is the case of the Ant Colony Optimization [52] and related improved versions such as [53], and Particle Swarm Optimization [54,55]. This last one with several different applications such as a vehicle routing problem [56–58] and engineering problems such as the design of near-field time delay equalizer metasurface [59], the Artificial Magnetic Conductor Surface [60], and for the design of a dielectric phase-correcting structure for antennas [61].

There are several MOEAs proposed in the literature, some of them are based on genetic algorithms and differ from each other on their replacement strategies; others are based on differential evolution. All of them use a population of current solutions P to generate offspring solutions O, combine them in $P \cup O$, and then employ a replacement strategy to generate a new population of solutions P'. In the sequence, we describe five MOEAs used in this work.

2.1.1. NSGAII

Non-dominated Sorting Genetic Algorithm-II [20] performs the replacement strategy considering Pareto Dominance and Crowding Distance selection, its major contribution. This selection evaluates how close solutions are to their neighbors, giving a better evaluation to large values, allowing them a better diversity in the population. Thus, NSGAII selects surviving solutions from $P \cup O$ first taking non-dominated solutions to compose P'. Two situations may occur: this set may be lower than or equal to the maximum population size or not. In the first case, it adds iteratively dominated solutions with higher Crowding Distance values until P' is complete. In the second case, NSGAII discards the solutions with lower Crowding values.

2.1.2. SPEA2

Differently from NSGAII, Strength Pareto Evolutionary Algorithm 2 [21] performs the replacement strategy considering Pareto Dominance and Strength values, which computes the relative difference between the number of other solutions that a particular solution dominates and those that it is dominated by. Higher values, meaning that a solution is more dominant, are better. As NSGAII, SPEA2 starts to fill P' population with non-dominated solutions and, if it is necessary to complete P', it selects dominated solutions with higher Strength value. In the case of more non-dominated solutions than allowed, SPEA2 does the same procedure as NSGAII.

2.1.3. IBEA

Indicator-Based Evolutionary Algorithm [25] performs a replacement considering a contribution of a particular solution to improve a specific quality indicator. This algorithm selects surviving solutions from $P \cup O$ by removing the ones which contribute less to the given quality indicator. Usually, Hypervolume is adopted as the quality indicator.

2.1.4. mIBEA

The Modified Indicator-Based Evolutionary Algorithm [43], based on IBEA, employs Hypervolume as the quality indicator. Different from its predecessor, which considers all solutions in $P \cup O$ to select solutions to compose P' based on the quality indicator contribution, this algorithm uses only non-dominated solutions from the union set, and then select the ones which contribute more. This algorithm works in the same way as IBEA after this. This modification improves the algorithm convergence and removes solutions with high-quality indicator contribution which are far away from the Pareto Front.

2.1.5. GDE3

Differently from the previously MOEAs, Generalized Differential Evolution 3 does not employ a crossover operator to generate offspring. Instead, this algorithm employs the differential evolution operator (DE) [62]. This operator generates offspring by combining more than three different parent solutions. This operator is performed until O is filled. The algorithm behaves like NSGAII on the further steps when generating the new population solution P'.

2.2. Selecting a MOEA

MOEAs do not generate a single final solution but a set of non-dominated solutions which are considered to be of the same quality. The performance comparison and selection of the best one among alternative algorithms can just be done using *quality indicators*. We can classify these indicators according to what they focus: convergence or diversity.

Convergence focuses on measuring the closeness of a given non-dominated solution set to the Pareto optimal front. Diversity focus on measuring how diverse the obtained solution set is along the Pareto optimal front [63].

Some quality indicators focus both on convergence and diversity: Hypervolume [64], Hyper-area Ratio (HR) [65], and Pareto Dominance Indicator (ER) [66]. Others focus on diversity: Uniform distribution (UD) [67] and a Ratio of Non-dominated Solutions (RNI) [67]. There are also other quality indicators, for example, Algorithm Effort (AE) [67], which focus on diversity but at the same time considers the computational time.

We can use quality indicators to determine which MOEA is the best for solving a given problem. However, due to the stochastic nature of these algorithms, we can just say an algorithm A is better than B according to a given quality indicator after running multiple trials of both, taking indicator averages and performing a statistical comparison. One way to reduce the overall computational effort is assigning an algorithm to do this task, in this case, a hyper-heuristic.

2.2.1. HHCF

Maashi et al. [6] proposed an online selection hyper-heuristic based on the Choice Function [7] named Hyper-Heuristic based on Choice Function. Their work aimed to select, one at a time, an LLH from a set H (with size n), and apply it along with g generations. To evaluate the performance of the LLHs, this approach uses a two-level-ranking system. First, each LLH is evaluated according to a group of m quality indicators, here composed by Hypervolume, RNI, UD, and AE.

Each quality indicator evaluates every LLH assigning a performance value to each of them. Then, LLHs are *ranked*, by the quality indicator, from the best performing LLH (rank 1) to the worst (rank n). At this point, a table containing all the quality indicators values for each LLH is created, a table with size $n * m$. This paper defines $RNI_{rank}(h)$ as the function that returns, for a given LLH h, his rank according to the RNI quality indicator. A second table ($Freq_{rank}$) is generated by computing how many times each LLH has the best value for each quality indicator. This is how HHCF summarizes several quality indicator preferences.

In order to select which LLHs to execute, HHCF selects a LLH that maximizes Equation (1), which is composed of an exploitation term f_1 and an exploration term f_2 weighted by a parameter α, a fixed parameter for this algorithm:

$$F(h_i) = \alpha f_1(h_i) + f_2(h_i) \tag{1}$$

The exploitation term f_1 is calculated by Equation (2). In this equation, n is the number of low-level heuristics, $Freq_{rank}(h_i)$ is the number of times that a given low-level heuristic h is the best one according to all quality indicators, and $RNI_{rank}(h_i)$ is the rank of the low-level heuristic according to the RNI quality indicator:

$$f_1(h_i) = 2 * (n + 1) - (Freq_{rank}(h_i) + RNI_{rank}(h_i)) \tag{2}$$

The exploration term f_2 is the computational waiting time (WT) that a given algorithm has waited inactive. In this present paper, due to the different computational effort demanded by the problems, we normalize f_2 using Equation (3):

$$f_2(h_i) = \frac{WT_{h_i}}{\sum_{j=0}^{n} WT_{h_j}} * 100 \quad (3)$$

Algorithm 1 illustrates how HHCF works. First, a random population of solutions is generated (Line 7) and used in the initialization process (Line 8). In this process, each $h \in H$ executes for g generations(Line 9), the current population Pop is updated and the values of quality indicators for Pop are computed and stored (Line 10). Afterwards, the algorithm continues with the process until the stopping criteria are met, by ranking each $h \in H$ (Line 12) and calculating Equation (2) (Line 13) and Equation (3) (Line 14). With this information, the LLH which maximizes Equation (1) is selected (Line 15) and used to generate solutions during g generations (Line 16). Finally, the new population Pop' is created using Pop and the offspring population (Line 17), and all the quality indicators are recalculated for h_i using Pop' (Line 18).

Algorithm 1: HHCF pseudocode.

```
1  Input:
2  Problem;
3  g—generations before evaluate an LLH;
4  H: set of LLHs {h_1, ..., h_i, ..., h_n};
5  α exploitation parameter;
6  begin
7      Generate a random population of solutions Pop;
8      Initialize components using H;
9      All h ∈ H uses Pop to generate Pop' during g generations;
10     Compute all quality indicators for all h ∈ H;
11     while A stopping criterion is not reached do
12         Compute Freq_rank and RNI_rank for all h ∈ H;
13         Equation (2) is computed for all h ∈ H;
14         Equation (3) is computed for all h ∈ H;
15         Select h_i according to Equation (1);
16         h_i executes for g generations and generates Pop';
17         Pop ← Pop' //acceptance criterion;
18         Compute all quality indicators for h_i;
19     end
20     return Pop
21 end
```

2.2.2. HHLA and HHRL

Learning Automata-based Multi-Objective Hyper-Heuristic with a Ranking scheme Initialization [3] implements a learning automata whose action is to select a LLH at each decision point, while the optimization problem is solved. There are two versions available of this algorithm: HHLA and HHRL. The only difference resides in the fact that HHRL employs an initialization process used in order to reduce the number of LLHs in the pool.

Algorithm 2 illustrates how both hyper-heuristics work. First, all of the initialization process is performed (Line 6). The algorithm continues while a stopping criterion is not reached; this hyper-heuristic applies the current LLH h_i to the current population (Pop) during g generations producing a new offspring (Pop'). In the following, Pop and Pop' are combined to generate the new current population Pop. In Line 11, this HH verifies whether it is time to switch or not to another LL: this is performed by verifying if is there an improvement in the Hypervolume value (compared to previous iterations). If it is the case, the current LLH keeps running, and, if not, the reinforcement learning scheme updates

the transition matrix P. Finally, another LLH is selected by the $\varepsilon-$RoulleteGreedy method from A considering the transition matrix P.

The $\varepsilon-$RoulleteGreedy method focuses on exploring different transition pairs by performing a given number of trials in order to get a better view of LLH pairwise performance at the early stage. Then, it becomes more and more greedy exploiting the accumulated knowledge.

As mentioned before, the only difference between HHLA and HHRL lies in the initialization method. Algorithm 3 describes this process. First, the method creates a random population of solutions (Line 5) and the transition matrix P (Line 6), which describes the selection probabilities of transitions between LLHs. If HHLA is being run (Line 7), all LLHs are allowed to execute. Otherwise, only allowed LLH is selected.

Algorithm 2: HHLA and HHRL pseudocode, adapted from [3].

```
1  Input:
2  Problem;
3  H: set of LLHs {h_1, ..., h_i, ..., h_n};
4  g fixed number of generations;
5  begin
6      [A, P, Pop, h_i] ← Initialization(H);
7      while A stopping criterion is not reached do
8          Pop' ← ApplyMetaHeuristic(h_i,Pop,g );
9          Pop ← Pop' //acceptance criterion;
10         //decide whether to switch to another metaheuristic;
11         if switch() then
12             LearningAutomataUpdateScheme(P);
13             h_i ← SelectMetaheuristic(P, A);
14         end
15     end
16     return Pop
17 end
```

Algorithm 3: HHLA and HHRL initialization pseudocode.

```
1  Input:
2  Problem;
3  H: set of LLHs {h_1, ..., h_i, ..., h_n};
4  begin
5      Pop ← Generate Random Population;
6      P ← Create transition Matrix;
7      if HHLA then
8          A ← H;
9      end
10     else if HHRL then
11         A ← SelectAllowedLLH(H);
12     end
13     h_i ← Select first LLH to run;
14     return A, P, Pop, h_i
15 end
```

For this purpose, the set of LLHs (H) is reduced in order to eliminate poor-performing LLHs. This works as follows: First, all LLHs are executed in sequence for a number of stages. Every time an LLH executes, HHRL computes the resulting population Hypervolume, computed using the same reference points. The scheme counts how many times an LLH becomes the best one in all stages. These counts are then used to determine which LLH should compose the allowed set A. LLHs with a performance worse than the average

are not allowed to compose A. The algorithm continues by selecting a current LLH h_i according to the $\varepsilon-$RoulleteGreedy and returning all the generated information.

2.2.3. MOABHH

Multi-Objective Agent-Based Hyper-Heuristic [28,29] is a hyper-heuristic designed as a multi-agent system. According to Wooldridge [68], an agent is "a computer system that is situated in some environment, and is capable of autonomous action in this environment in order to meet its objectives".

This hyper-heuristic is designed to consider LLHs and quality indicators as agents in one election, which means that LLHs are candidates to be voted by quality indicator agents (the voters). The election happens each g generations, and the outcome tells us which LLH is performing better. Then, the election winners generate more offspring after the election.

MOABHH differs from HHCF, HHRL, and HHLA regarding how the current population is processed by the LLHs. In this algorithm, all LLH executes in parallel acting on a share of the main population and generating offspring in the same generation. This is performed by splitting the main population into subpopulations according to election outcomes where the best LLH receives a bigger subpopulation and can generate more offspring. In the beginning, this main population is equally split into subpopulations, each one processed by a different *LLH agent*, which will receive a sub-population, generate new offspring solutions, and find the surviving solutions.

When it is the election time, all quality indicators' agents (voters) evaluate each LLH agent (candidate), rank them, and send their rank to the *HH agent*, which is responsible for taking all votes and processing them according to an election method in order to generate an election outcome. In our approach, we use the Copeland [69] voting method, in which candidates are ordered by the number of pairwise victories, minus the number of pairwise defeats. The election outcome is used by the *HH agent* to increase the participation in generating offspring to the election winners and decreasing it to the losers in the next cycles.

Algorithm 4 details MOABHH steps. First, all agents, components, and global variables are initialized (line 3). A random population of solutions is generated (line 4). The execution continues creating a *Participation* array that is responsible for determining how many solutions each LLH can generate per generation. At this time, this array is created uniformly, assigning the same participation in generating offspring at the begging of the search.

The algorithm continues by splitting the population into subpopulations according to the *Participation* array. Then, all LLH generates offspring and updates the main population in parallel in a synchronized task.

Each g generations the voting process then starts, and the *Voter* agents evaluate the solutions produced by LLHs, rank them according to their preferences (line 18), and send them to the *HH* agent. After that, the *HH* agent calculates the social ranking according to the Copeland voting method (line 19) and assigns a bigger participation in the population for the election winner and a lower one for election losers (line 20).

Algorithm 4: MOABHH pseudocode.

```
Input:
Problem;
g—generations before evaluate an LLH;
H: set of LLHs {h_1, ..., h_i, ..., h_n};
begin
    Initialize agents and components using H;
    Generate a random population of solutions Pop;
    Participation ← Uniformly share the number of solutions to generate;
    while A stopping criterion is not reached do
        gen ← 0;
        while gen < g do
            Split Pop into subpopulations according to the Participation;
            LLH Agents execute for one generation;
            LLH Agents update the main population;
            gen ← gen + 1;
        end
        if It is election time then
            Voter agents evaluate LLH agents outcomes and vote;
            HH agent performs the voting method;
            HH agent uses the election outcome to update the Participation;
        end
    end
    return Pop
end
```

2.3. Real-World Multi-Objective Problems

Over the years, several artificially constructed test problems have been proposed to compose benchmarks for evaluating meta-heuristics. These problems offer many advantages over real-world problems for the purpose of general performance testing [70], by allowing users to compare the results of their algorithms (regarding effectiveness and efficiency) with others, over a spectrum of algorithms' instantiations [46].

In the literature, one can find several of these MOP benchmarks, such as WFG, DTLZ [27], and UF [12]. However, even if an algorithm has successfully solved these problems, this does not guarantee effectiveness and efficiency in solving real-world problems [46].

In terms of multi-objective hyper-heuristics, researchers have been using both benchmarks and real-world applications. However, few studies consider more than one real-world problem. This choice can, in fact, diminish the accuracy of evaluating hyper-heuristics on cross-domain applications.

Table 1 presents the eighteen real-world problems that we have used in this work. Ten of them were picked from *Black Box Optimization Competition* (https://www.ini.rub.de/PEOPLE/glasmtbl/projects/bbcomp/downloads/realworld-problems-bbcomp-EMO-2017.zip, accessed date 10 April 2021) [38], a group of problems created/selected for the 9th International Conference on Evolutionary Multi-Criterion Optimization (EMO'2017) (http://www.emo2017.org/, accessed date 10 April 2021). We included the CrashWorthiness problem due to the fact it was already considered in previous papers for all the hyper-heuristics. The problems Water, Machining, and CarSideImpact were studied using MOABHH, but not for HHCF, HHLA, and HHRL. The other four problems were selected from the optimization literature and picked from the jMetal framework [71], the framework used by all studied hyper-heuristics. In the table, their number of objectives, variables, and constraints are detailed. All the problems are in continuous space.

Table 1. A brief description real-world multi-objective problems containing the number of objectives, variables constraints, and source.

ID	Problem Name	Objs.	Vars.	Const.	Source
P01	Water	5	3	7	[33]
P02	Machining	4	3	3	[35]
P03	CarSideImpact	3	7	0	[34]
P04	CrashWorthiness	3	5	0	[19]
P05	FourBarTruss	2	4	0	[39]
P06	Golinski	2	7	11	[40]
P07	Quagliarella	2	16	0	[41]
P08	Poloni	2	2	0	[42]
P09	Vibrating Platform Design	2	5	0	[38,72]
P10	Optical Filter	2	11	0	[38,73]
P11	Welded Beam Design	2	4	0	[38,74]
P12	Disk Brake Design	2	4	0	[38,75]
P13	Heat Exchanger	2	16	0	[38]
P14	Hydro Dynamics	2	6	0	[38]
P15	Area Under Curve	2	10	0	[38]
P16	Kernel Ridge Regression	2	5	0	[38]
P17	Facility Placement	2	20	0	[38]
P18	Neural Network Controller	2	24	0	[38]

3. Methodology

We set up the four studied hyper-heuristic controlling five LLH (GDE3, IBEA, NSGAII, SPEA2, and mIBEA) to solve the eighteen continuous real-world optimization problems presented in Table 1. Different configurations were employed for these problems. In particular, the number of generations and population size were slightly different for P17-P18. This was due to the high computational effort demanded in these applications, which takes almost three months for experiments using the same setup considered for the problems P01–P16.

Table 2 presents all these parameter used in our experiments. We set up HHLA and HHRL parameters according to [3], MOABHH according to [32] and HHCF according to [6] for P01–P16. For P17 and P18, we set MOABH $g = 1, \beta = 0.5$ and HHLA and HHRL $g = 1$. For these two problems, parameters were defined empirically.

All the original hyper-heuristics were designed to work with populational genetic algorithms. They have specific procedures to manipulate the current population of solutions. In this comparison, we add two other MOEAs (mIBEA and GDE3), which are modeled the same as the three others. For this reason, it does not demand deep changes in the compared hyper-heuristics, changes that would define new hyper-heuristics. Thus, algorithms such as MOEA/D [76] and SMPSO [55] cant be considered without proposing four new algorithms.

For genetic algorithms, we followed [3,6]. For GDE3, we followed [32]. We analyzed the performance comparing both hyper-heuristic results with the five single meta-heuristic results. All MOEAs used are implemented by jMetal 5.7 [71]. Each experiment is repeated 30 times.

Table 2. Parameters used in experiments.

Problems	Param	IBEA	GDE3	mIBEA	NSGA-II	SPEA2	HHCF	HHLA	HHRL	MOABHH
All	Crossover type	SBX	-	SBX	SBX	SBX	SBX	SBX	SBX	SBX
	Crossover prob.	0.9	-	0.9	0.9	0.9	0.9	0.9	0.9	0.9
	Crossover dist.	20	-	20	20	20	20	20	20	20
	Mutation type	Poly	Poly	Poly	Poly	Poly	Poly	Poly	Poly	Poly
	Mutation prob.	$\frac{1}{vars}$	$\frac{1}{vars}$	$\frac{1}{vars}$	$\frac{1}{vars}$	$\frac{1}{vars}$	$\frac{1}{vars}$	$\frac{1}{vars}$	$\frac{1}{vars}$	$\frac{1}{vars}$
	Mutation dist.	20	20	20	20	20	20	20	20	20
	DE type	-	$rand/1/bin$	-	-	-	$rand/1/bin$	$rand/1/bin$	$rand/1/bin$	$rand/1/bin$
	DE CR	-	0.2	-	-	-	0.2	0.2	0.2	0.2
	DE F	-	0.2	-	-	-	0.2	0.2	0.2	0.2
	DE K	-	0.5	-	-	-	0.5	0.5	0.5	0.5
	HHCF α	-	-	-	-	-	30	-	-	-
	Decision Points	-	-	-	-	-	-	25	25	-
	HHRL/HHLA m	-	-	-	-	-	-	2	2	-
	HHRL/HHLA α	-	-	-	-	-	-	0.1	0.1	-
	HHRL/HHLA β	-	-	-	-	-	-	3	3	-
P01-P16	Generations	1000	1000	1000	1000	1000	1000	1000	1000	1000
	Pop Size	100	100	100	100	100	100	100	100	100
	HHCF g	-	-	-	-	-	25	-	-	-
	HHRL/HHLA g	-	-	-	-	-	25	-	-	-
	MOABHH g	-	-	-	-	-	-	-	-	50
	MOABHH β	-	-	-	-	-	-	-	-	1
P17-P18	Generations	50	50	50	50	50	50	50	50	50
	Pop Size	30	30	30	30	30	30	30	30	30
	HHCF g	-	-	-	-	-	1	-	-	-
	HHRL/HHLA g	-	-	-	-	-	1	-	-	-
	MOABHH g	-	-	-	-	-	-	-	-	1
	MOABHH β	-	-	-	-	-	-	-	-	0.5

We employed Hypervolume and IGD+ averages obtained from the 30 executions. First, for each problem, we joined all results obtained by all algorithms, found the nadir point, necessary for Hypervolume calculation, and took the non-dominated set in order to generate the *known Pareto Front* (PF_{known}), necessary for IGD+ calculation. Then, we calculated the quality indicator averages and compared them using Kruskal–Wallis as the statistical test with a confidence level of 95%. In order to perform this, we first identify which algorithm has the best average according to the quality indicator. Thus, all the other algorithms are compared to the best, generating a set of p-values. We define an algorithm tied statistically with the best when a given p-value is superior to the significance level, which in this case is 0.05.

4. Experimental Results

In this section, we present the empirical results and our analysis. Tables 3 and 5 present, respectively, averages for Hypervolume and IGD+. For each problem, the algorithms results were submitted to a Kruskal–Wallis statistical test where the following hypothesis was answered:

Hypothesis 1. *Considering a quality indicator (Hypervolume or IGD+), is a given algorithm output equivalent to the algorithm that has the best output?*

In these tables, we highlighted (in grey) the algorithm with the best average output; moreover, considering the statistic test outcomes, we represented in bold those algorithms whose outputs are equivalent to the best one.

4.1. Hypervolume Analysis

Table 3 presents Hypervolume averages. From the experiments, we could conclude the following:

- If we compare the results obtained *just by the five MOEAs*, we could see that: (i) GDE3 obtained the highest results on five problems (P01, P04, P14, P16, and P17); (ii) IBEA was the best one on four problems (P02, P05, P07, and P13); (iii) NSGAII has the best average on four problems (P06, P10, P15, and P18); (iv) SPEA2 has the best average on five problems (P03, P08, P09, P11, and P12); and (v) mIBEA did not get the best results in any experiment.
- If we consider *just the four hyper-heuristics*, we have: (i) MOABHH has higher averages on nine problems (P01, P04, P07, P08, P10, P12, and P15–P17); (ii) HHLA has its best results on three problems (P05, P14, and P18); (iii) HHRL performs better than others on five problems (P02, P03, P09, P11, and P13); (iv) HHCF is the best algorithm just in P06.
- Finally, if we compare the results obtained by *all nine algorithms*, GDE3 has higher Hypervolume values on four problems (P04, P14, P16, and P17), IBEA has better averages on three problems (P05, P07, and P13), NSGAII has the best result on three problems (P06, P10, and P18), SPEA2 on three problems (P08, P09, and P11), MOABHH has better averages on three problems (P01, P12, and P15), and HHRL in two of them (P02 and P03). Moreover, mIBEA (among MOEAs), HHLA, and HHCF (both among HHs) did not excel in any problem.

Finally, it is interesting to compare the results obtained by HHs with individual MOEAs. Table 4 presents a summary of the statistical pairwise comparison between HHs and individual MOEAs considering Hypervolume averages. An x in a certain cell means that the corresponding HH could not achieve a result as good as the one obtained by a particular MOEA, considering the Hypervolume indicator. First of all, we can notice that all HHs were overcome in problems P17 and P18. Moreover, MOABHH could not achieve statistically tied results with the best algorithms in three other problems (P10, P11, and P14), HHLA was bet in six other problems (P04, P06, P08, P09, P10, and P12), HHRL could not achieve the best result in two other problems (P06 and P10), and HHCF did not get good results in six other problems (P05, P07, P08, P09, P11, and P16).

Table 3. Hypervolume averages considering 30 executions. Highlighted values are the best Hypervolume values among all nine algorithms. **Bold** values are tied statistically with the best value.

Problem	GDE3	IBEA	NSGAII	SPEA2	mIBEA	MOABHH	HHLA	HHRL	HHCF
P01	**2.92912×10^{24}**	2.13470×10^{24}	2.74801×10^{24}	2.79395×10^{24}	2.65003×10^{24}	**2.92954×10^{24}**	**2.91106×10^{24}**	**2.91360×10^{24}**	**2.85991×10^{24}**
P02	**1.20988×10^{1}**	**1.25672×10^{1}**	**1.17447×10^{1}**	**1.24937×10^{1}**	**1.25351×10^{1}**	**1.22389×10^{1}**	**1.25694×10^{1}**	**1.25745×10^{1}**	**1.22089×10^{1}**
P03	**3.40403×10^{1}**	3.28857×10^{1}	3.33919×10^{1}	3.42588×10^{1}	3.28639×10^{1}	**3.39933×10^{1}**	**3.42089×10^{1}**	**3.43667×10^{1}**	**3.43608×10^{1}**
P04	**4.12497×10^{1}**	3.91635×10^{1}	4.10191×10^{1}	**4.10782×10^{1}**	3.91409×10^{1}	**4.11873×10^{1}**	4.08916×10^{1}	**4.11610×10^{1}**	**4.11795×10^{1}**
P05	4.29245×10^{1}	**4.31194×10^{1}**	4.29347×10^{1}	4.29918×10^{1}	**4.31192×10^{1}**	**4.30434×10^{1}**	**4.31166×10^{1}**	**4.31071×10^{1}**	4.30204×10^{1}
P06	**1.83792×10^{6}**	1.83368×10^{6}	**1.83826×10^{6}**	1.83533×10^{6}	1.83376×10^{6}	**1.83721×10^{6}**	1.83526×10^{6}	1.83597×10^{6}	**1.83772×10^{6}**
P07	4.60452×10^{0}	**4.65843×10^{0}**	4.56779×10^{0}	4.61121×10^{0}	**4.64988×10^{0}**	**4.64656×10^{0}**	**4.64647×10^{0}**	**4.63820×10^{0}**	4.58697×10^{0}
P08	3.68057×10^{2}	3.62909×10^{2}	3.68039×10^{2}	**3.68161×10^{2}**	3.65549×10^{2}	**3.68023×10^{2}**	3.65906×10^{2}	**3.67855×10^{2}**	3.67552×10^{2}
P09	8.86141×10^{-1}	8.84318×10^{-1}	8.86664×10^{-1}	**8.86948×10^{-1}**	8.83240×10^{-1}	**8.85986×10^{-1}**	8.85831×10^{-1}	**8.86501×10^{-1}**	8.85131×10^{-1}
P10	**7.65400×10^{-1}**	7.58649×10^{-1}	**7.68186×10^{-1}**	7.68104×10^{-1}	7.56017×10^{-1}	7.68160×10^{-1}	7.64903×10^{-1}	7.66534×10^{-1}	**7.63534×10^{-1}**
P11	7.51140×10^{-1}	7.51382×10^{-1}	7.52204×10^{-1}	**7.53349×10^{-1}**	7.47803×10^{-1}	7.51807×10^{-1}	**7.52894×10^{-1}**	**7.53102×10^{-1}**	7.52468×10^{-1}
P12	7.39087×10^{-1}	7.30252×10^{-1}	7.39380×10^{-1}	**7.40053×10^{-1}**	7.30268×10^{-1}	**7.40062×10^{-1}**	7.36477×10^{-1}	**7.38370×10^{-1}**	**7.40020×10^{-1}**
P13	4.51948×10^{-1}	**4.60981×10^{-1}**	**4.52136×10^{-1}**	**4.51396×10^{-1}**	**4.55251×10^{-1}**	**4.60470×10^{-1}**	**4.53651×10^{-1}**	**4.60516×10^{-1}**	**4.49730×10^{-1}**
P14	**6.56094×10^{-1}**	6.52222×10^{-1}	**6.54562×10^{-1}**	**6.54886×10^{-1}**	6.47015×10^{-1}	6.55449×10^{-1}	**6.55734×10^{-1}**	**6.55663×10^{-1}**	**6.52926×10^{-1}**
P15	**9.88099×10^{-1}**	9.80715×10^{-1}	**9.89134×10^{-1}**	**9.89115×10^{-1}**	**9.88014×10^{-1}**	**9.89425×10^{-1}**	**9.89091×10^{-1}**	**9.88244×10^{-1}**	**9.87321×10^{-1}**
P16	**3.78647×10^{-1}**	5.19911×10^{-2}	**2.99906×10^{-1}**	**2.07564×10^{-1}**	1.18439×10^{-1}	**2.40525×10^{-1}**	**1.78549×10^{-1}**	**2.14554×10^{-1}**	1.43599×10^{-1}
P17	**9.07392×10^{-1}**	8.68258×10^{-1}	**9.03361×10^{-1}**	**9.01765×10^{-1}**	8.94593×10^{-1}	8.94496×10^{-1}	8.94128×10^{-1}	8.77702×10^{-1}	8.92694×10^{-1}
P18	**5.28786×10^{-1}**	5.04137×10^{-1}	**5.29036×10^{-1}**	4.97061×10^{-1}	**5.00425×10^{-1}**	4.93037×10^{-1}	4.96037×10^{-1}	4.09329×10^{-1}	4.78611×10^{-1}

Table 4. Summary of the statistical pairwise comparison between HHs and individual MOEAs considering Hypervolume averages. An x means that the HH could not achieve a result as good as the one obtained by an MOEA.

Problem	MOABHH	HHLA	HHRL	HHCF
P01				
P02				
P03				
P04		x		
P05				x
P06		x	x	
P07				x
P08		x		x
P09		x		x
P10	x	x	x	
P11	x			x
P12		x		
P13				
P14	x			
P15				
P16				x
P17	x	x	x	x
P18	x	x	x	x
Total	**5**	**8**	**4**	**8**

4.2. IGD+ Analysis

Table 5 presents IGD+ averages. From the experiments, we could conclude the following:

- If we compare the results obtained *just by the five MOEAs*, we could see that: (i) GDE3 has better averages on five problems (P01, P04, P14, P16, and P17); (ii) IBEA excelled in four problems (P7, P11, P13, and P15); (iii) NSGAII has obtained good results in two problems (P06 and P18); (iv) SPEA2 has obtained the best IGD+ values for the five problems (P02, and P8–P11); and (v) mIBEA has obtained the best results in two problems (P03 and P05).
- If we consider *just the four hyper-heuristics,* we have: (i) MOABHH excels in seven problems (P01, P07, P08, P10, P12, P14, and P17); (ii) HHLA performs better in four problems (P05, P11, P15, and P18); (iii) HHRL performs better in six problems (P02–P04, P09, P11, and P13); and (iv) HHCF has good results in two problems (P06 and P16).
- Finally, if we compare the results obtained by *all nine algorithms*, GDE3 is the best algorithm on three problems (P04, and P16–P17), IBEA performs better on three problems (P07, P11, and P15), NSGAII on 2 problems (P06 and P18), SPEA2 excels on four problems (P02, P08, P09, and P11), mIBEA just on P05, MOABHH performs better on four problems (P01, P10, P12, and P14), and HHRL also in two problems (P03 and P13). Moreover, HHLA and HHCF (both among HHs) did not excel in any problem.

Table 5. IGD+ averages considering 30 executions. Highlighted values are the best IGD+ values among all nine algorithms. **Bold** values are tied statistically with the best value.

Problem	GDE3	IBEA	NSGAII	SPEA2	mIBEA	MOABHH	HHLA	HHRL	HHCF
P01	**0.03550**	0.13713	0.05409	0.05171	0.07120	**0.03529**	**0.03790**	**0.03825**	**0.04425**
P02	**0.04721**	**0.04603**	**0.05449**	**0.04067**	**0.04650**	**0.04603**	**0.04294**	**0.04138**	**0.04741**
P03	0.02962	**0.01917**	0.03327	0.02179	**0.01914**	0.02923	**0.01942**	**0.01873**	**0.02108**
P04	**0.00346**	0.02048	0.00451	**0.00510**	0.02017	**0.00422**	0.00559	**0.00387**	**0.00401**
P05	0.00081	**0.00022**	0.00075	0.00063	**0.00021**	**0.00056**	**0.00023**	**0.00026**	0.00049
P06	**0.00027**	0.00106	**0.00022**	0.00108	0.00106	**0.00042**	0.00081	0.00066	**0.00032**
P07	0.00056	**0.00030**	0.00147	**0.00044**	**0.00045**	**0.00039**	**0.00045**	0.00061	0.00100
P08	0.00048	0.00511	0.00050	**0.00027**	0.00388	0.00050	0.00316	**0.00058**	0.00110
P09	0.00156	0.00105	0.00110	**0.00057**	0.00087	**0.00072**	**0.00074**	**0.00062**	**0.00078**
P10	0.00206	**0.00306**	0.00106	**0.00056**	0.00366	**0.00038**	**0.00129**	**0.00084**	0.00206
P11	0.00230	**0.00102**	0.00194	**0.00102**	0.00177	0.00186	**0.00109**	**0.00109**	0.00165
P12	0.00146	0.00148	0.00133	**0.00069**	0.00148	**0.00067**	**0.00088**	**0.00077**	**0.00072**
P13	0.00348	**0.00188**	0.00339	**0.00331**	**0.00242**	**0.00159**	**0.00292**	**0.00149**	0.00417
P14	0.00056	**0.00646**	0.00382	0.00338	**0.01609**	**0.00007**	**0.00011**	0.00016	0.00673
P15	0.58012	**0.05165**	0.56817	0.58503	0.51695	0.60273	0.57720	0.60337	0.60657
P16	**0.15856**	0.27179	**0.19379**	**0.25513**	**0.19813**	**0.24176**	**0.24395**	0.22619	**0.20020**
P17	**0.02195**	0.14605	**0.03079**	**0.03281**	0.04614	0.04976	0.05849	0.06777	0.06026
P18	**0.10275**	**0.11558**	**0.09524**	**0.12183**	**0.11506**	0.12440	0.12341	0.21049	0.13727

Finally, as done in Section 4.1, it is also interesting to compare the results obtained by HHs with individual MOEAs. Table 6 presents a summary of the statistical pairwise comparison between HHs and individual MOEAs considering IGD+ averages. An x in a certain cell means that the corresponding HH could not achieve a result as good as the one obtained by a particular MOEA, considering the IGD+ indicator.

Table 6. Summary of the statistical pairwise comparison between HHs and individual MOEAs considering IGD+ averages. An x means that the HH could not achieve a result as good as the one obtained by a MOEA.

Problem	MOABHH	HHLA	HHRL	HHCF
P01				
P02				
P03	x			
P04		x		
P05				x
P06		x	x	
P07			x	x
P08	x	x		x
P09				
P10				x
P11	x			x
P12				
P13				x
P14			x	
P15	x	x	x	x
P16			x	
P17	x	x	x	x
P18	x	x	x	x
Total	**6**	**6**	**7**	**9**

First of all, we can notice that all HHs were overcome in problems P15, P17, and P18. Moreover, MOABHH could not achieve statistically tied results with the best algorithms in three other problems (P03, P08, and P11), HHLA was also bet in three other problems (P04, P06, P08), HHRL could not achieve the best result in four other problems (P06, P07, P14, and P16), and HHCF did not get good results in six other problems (P05, P07, P08, P10, P11, and P13).

4.3. Hyper-Heuristics Analysis

Utilization of Low-Level Meta-Heuristics

In this section, we address the following issue: how much a single LLH is chosen by a particular hyper-heuristics. Figure 1 graphically presents this usage: for MOABHH, it represents the percentage of participation in generating offspring along with the search. On the other hand, for HHRL, HHLA, and HHCF, the figure presents the percentage of times that each LLH was chosen. The data consider all problem instances, each of them running 30 times.

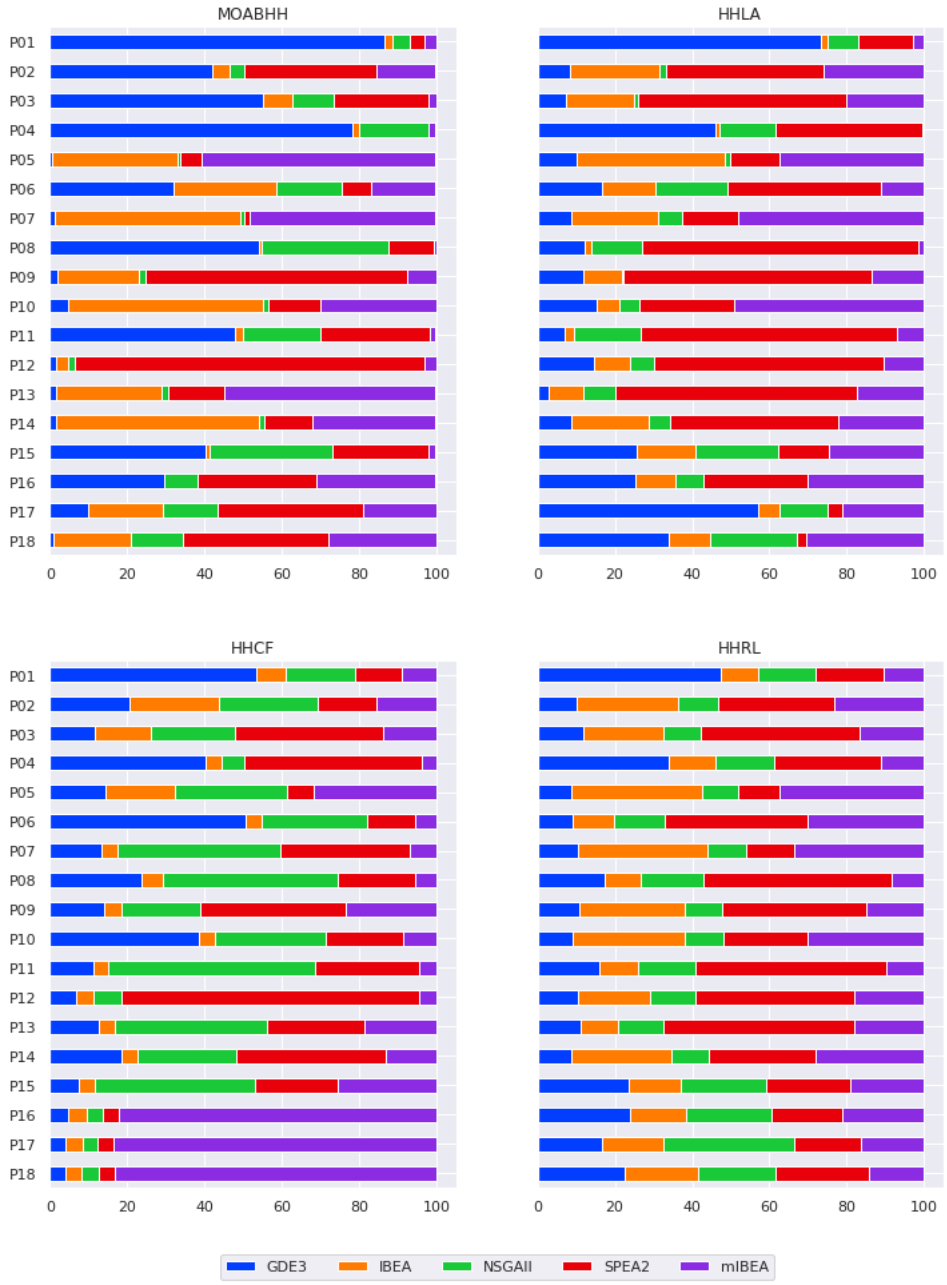

Figure 1. Utilization rate for the four hyper-heuristics.

The particular behavior of each HH may be found by analyzing Figure 1. For example, if we consider problem P03, one may notice that MOABHH has chosen more times GDE3 (blue) and SPEA2 (red), while HHLA chose more often SPEA2. On the other hand, HHRL and HHCF have chosen LLHs more uniformly. Table 7 presents a summarized evaluation of this analysis, where we classified HHs'c behavior into four classes: (i) *One Elitist*: problems

where one LLH is clearly selected more times than any other, i.e. more than 50%; (ii) *Two Elitist*: problems where two LLHs are privileged, i.e. each LLH selected more than 40% ; (iii) *Three Elitist*: problems where three LLHs are selected more times than others, i.e., each LLH selected more than 30%; (iv) *Not Elitist*: when there is no clear LLH preference.

Table 7. How elitist HHs are on selection LLHs.

	One Elitist	Two Elitist	Three Elitist	Not Elitist
MOABHH	P01, P03, P04, P05, P09, P12	P02, P07, P08, P10, P11, P13, P14	P15, P16	P06, P17, P18
HHLA	P01, P03, P07, P08, P09, P11, P12, P13, P17	P04, P05, P10	P14, P16, P18	P02, P06, P15
HHCF	P01, P06, P11, P12, P16, P17, P18	P04, P07	P05, P08, P10, P15	P02, P03, P09, P13, P14
HHRL	P01, P08, P11, P12, P13	P04, P05, P06, P07, P09	P02, P03, P10, P14	P15, P16, P17, P18

We can identify HHLA and HHCF with more problems classified as *One Elitist*, while MOABHH and HHRL had more problems in *Two Elitist* category. For *Three Elitist*, HHLA, HHCF, and HHRL had four problems classified while MOABHH had three. Considering all elitist classified problems (*One Elitist* + *Two Elitist* + *Three Elitist*), we can identify all HH behaving in a similar way.

4.4. Generality Analysis

In order to perform a cross-domain evaluation of algorithms, we followed [77] and generated the average and aligned Friedman ranking considering both Hypervolume and IGD+ values. These rankings consider which position each algorithm takes on each problem. We also concatenated Hypervolume (Table 3) and IGD+ (Table 5) tables, both with 18 lines of data, in order to create a new table of mixed quality indicators with 36 lines of data. As best Hypervolume and IGD+ values are, respectively, the highest and the lowest ones, we have used $\overline{IGD+} = 1 - IGD+$ values instead.

Table 8 presents the statistical evaluation. In this table, smaller statistical values are considered as better.

Considering just MOEA results, SPEA2 is the best single MOEA according to almost all statistical scores. The second best is GDE3, while IBEA and mIBEA are the worst-performing MOEAs according to this cross-domain analysis, with mIBEA performing 'slightly' better than IBEA.

Table 8. Friedman Ranking and Aligned Friedman Rank of the algorithms for Hypervolume and IGD+. Highlighted values are the best values among all nine algorithms.

Algorithm	Friedman Hypervolume Rank	Friedman Hypervolume Aligned Rank	Friedman IGD+ Rank	Friedman IGD+ Aligned Rank	Friedman Mixed Rank	Friedman Mixed Aligned Rank
GDE3	4.22	69.72	4.88	77.05	4.55	145.08
IBEA	6.61	109.33	5.78	97.61	6.19	208.22
NSGAII	4.5	83.56	5.22	78.27	4.86	160.69
SPEA2	4.17	69.94	4.56	72.27	4.36	142.11
mIBEA	6.72	108.72	5.44	86.22	6.08	196.77
MOABHH	3.77	60.77	4.05	73.34	3.91	131.00
HHLA	4.94	78.44	4.88	81.34	4.25	156.83
HHRL	4.11	66.44	4.38	78.34	5.86	141.75
HHCF	5.94	86.55	5.77	89.05	5.47	180.02

Considering all of the nine studied algorithms, *MOABHH is the algorithm which performs better in a cross-domain perspective*, as highlighted (in grey) in Table 8. HHRL is the second-best except considering mixed average ranking values (5th column of the table): in this specific case, HHLA is considered the second-best algorithm.

Another interesting result is that it is not the case that an HH always gets better results than the LLHs that it is composed of; one can notice that, in general, SPEA2 and GDE3 got better statistical values when compared both to HHLA and HHCF.

4.5. Discussion

The cross-domain tests illustrated that mIBEA is one of the worst poor-performing algorithms for this group of problems, while GDE3 is one of the top-performing algorithms. Thus, the inclusion of mIBEA and GDE3 in the LLH pool increases the difficulty of choosing the best algorithm for hyper-heuristics. In the previous studies, mainly three algorithms were considered in the LLH pool: IBEA (one of the worst algorithms in our study), SPEA2 (the best one in this study), and NSGAII.

MOABHH and HHRL performed quite well, showing clearly superior results when compared to HHLA and HHCF. Both MOABHH and HHRL removed the poor performing LLHs at the beginning of the search, letting the best LLH run more time than others. HHLA and HHCF, however, kept trying poor-performing LLHs. HHRL removed the poor performing LLHs right away, without giving a proper chance to them during its initialization process, but those LLHs might have performed well in the later stages of the search process. On the other hand, MOABHH has kept all the algorithms running in parallel and removing a percentage of the offspring generation from the worse algorithms. This increases the MOABHH capability of exploring the search and avoiding the chance of removing an LLH with potential good performance in the later stages from the LLH pool in the beginning of the execution.

5. Conclusions

In this study, we investigated reportedly the top four online selection hyper-heuristics across eighteen real-world optimization problems. The hyper-heuristics controlled and mixed a set of five low-level MOEAs to produce improved trade-off solutions. The performance of algorithms was also evaluated with a larger set of MOEAs.

To the best of the authors' knowledge, this work is the first one which addresses the problem of using real-world problem instances for cross-domain performance evaluation of hyper-heuristics. In particular, we addressed in this paper the following issues: (i) an evaluation of four state-of-the-art online hyper-heuristics (MOABHH, HHRL, HHLA, and HHCF) using exclusively real-world problems; (ii) a harder selection task for these four hyper-heuristics by increasing the number of Low-Level heuristics used: in our work, we used five LLHs, whereas, in previous publications, the number of LLHs used were 3 or 4; (iii) a cross-domain tested formed by eighteen real-world optimization problems (presented in Table 1) in order to evaluate multi-objective hyper-heuristics, which gives a more realistic overview of their performance.

As expected, the empirical results showed that individual MOEAs deliver different performances on different problems, making those real-world problem instances very useful for cross-domain performance evaluation of hyper-heuristics. Moreover, our results showed that hyper-heuristics have a better cross-domain performance than single meta-heuristics. This means that, when a new multi-objective optimization problem must be solved, hyper-heuristics are excellent candidates, reducing the user's effort to run repeatedly several meta-heuristics with different parameters settings in order to get a solution. In particular, MOABHH turned out to be the best algorithm delivering the best overall cross-domain performance, beating the other state-of-the-art hyper-heuristics with respect to two quality indicators: IGD+ and Hypervolume.

As future work, these hyper-heuristics will be studied across various applications in the discrete multi-objective optimization domain, such as Search-Based Software Engineering Problems [78] and exploring the potential of these algorithms on helping to improve the current pandemic, in terms of diagnosis and treatment [79] and drug discovery [80]. Many objective problems also impose a challenge for researchers and practitioners: some many-objective approaches were recently proposed in the literature. Another research direction would be to study the performance of the top-performing online learning selection hyper-heuristics across problems with more than eight objectives, varying their LLHs.

Author Contributions: Conceptualization: all authors; Formalization, implementation and experiments: V.R.d.C.; Supervision: E.Ö. and J.S.S.; Writing and proofreading the paper: all authors. All authors have read and agreed to the published version of the manuscript.

Funding: This study was partially supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brasil (CAPES)—grant code 001. Vinicius Renan de Carvalho was also supported by CNPq, Brazil, Grant No. 140974/2016-4.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: MOABHH source code can be found in https://github.com/vinixnan/MOABHH. HHRL, HHLA, and HHCF code can be found in https://github.com/vinixnan/MOHH-LARILA-Real. All employed problems can be found in https://github.com/vinixnan/jMetalHyperHeuristicHelper, all accessed on 6 August 2021.

Acknowledgments: We thank Wenwen Li by providing MOHH-RILA and mIBEA code.

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

Abbreviations

AE	Algorithm Effort
DE	Differential Evolution
DTLZ	Deb, Thiele, Laumanns, and Zitzler's Benchmark
GDE3	Generalized Differential Evolution 3
HH	Hyper-Heuristic
HHCF	Maashi's Choice Function Hyper-Heuristic
HHLA	MOHH-RILA instance with no LLH removal during the initialization
HHRL	MOHH-RILA instance with LLH removal during the initialization
HR	Hypervolume Ratio
IBEA	Indicator Based Evolutionary Algorithm
IGD	Inverted Generational Distance
IGD+	Inverted Generational Distance Plus
LLH	Low-Level Heuristic
MAB	Multi-Armed Bandit
mIBEA	Modified Indicator Based Evolutionary Algorithm
MOABHH	Multi-Objective Agent-Based Hyper-Heuristic
MOGA	Multi-Objective Genetic Algorithm
MOEA	Multi-Objective Evolutionary Algorithm
MOHH-RILA	Learning Automata-based Multi-Objective Hyper-Heuristic with a ranking scheme initialization
MOP	Multi-Objective Optimization Problem
NSGAII	Non-Dominated Sorting Genetic Algorithm II
PF	Pareto Front
RNI	Ratio of Non-Dominated Solution
SPEA2	Strength Pareto Evolutionary Algorithm 2
UD	Uniform Distribution
WFG	Walking Fish Group's benchmark
ZDT	Zitzler, Deb, and Thiele's Benchmark

References

1. Burke, E.K.; Gendreau, M.; Hyde, M.; Kendall, G.; Ochoa, G.; Özcan, E.; Qu, R. Hyper-heuristics: A survey of the state of the art. *J. Oper. Res. Soc.* **2013**, *64*, 1695–1724.
2. Drake, J.H.; Kheiri, A.; Özcan, E.; Burke, E.K. Recent Advances in Selection Hyper-heuristics. *Eur. J. Oper. Res.* **2020**, *285*, 405–428.
3. Li, W.; Özcan, E.; John, R. A Learning Automata-Based Multiobjective Hyper-Heuristic. *IEEE Trans. Evol. Comput.* **2019**, *23*, 59–73.
4. Burke, E.; Hyde, M.; Kendall, G.; Ochoa, G.; Özcan, E.; Woodward, J. A classification of hyper-heuristic approaches. In *Handbook of Metaheuristics*; Springer: Boston, MA, USA, 2010; Volume 146, pp. 449–468.
5. Burke, E.K.; Hyde, M.R.; Kendall, G.; Ochoa, G.; Özcan, E.; Woodward, J.R. A Classification of Hyper-Heuristic Approaches: Revisited. In *Handbook of Metaheuristics*; Springer International Publishing: Cham, Switzerland, 2019; Chapter 14, pp. 453–477.

6. Maashi, M.; Özcan, E.; Kendall, G. A multi-objective hyper-heuristic based on choice function. *Expert Syst. Appl.* **2014**, *41*, 4475–4493.
7. Cowling, P.I.; Kendall, G.; Soubeiga, E. A Hyperheuristic Approach to Scheduling a Sales Summit. In Proceedings of the Third International Conference on Practice and Theory of Automated Timetabling, Konstanz, Germany, 16–18 August 2000; pp. 176–190.
8. Li, K.; Fialho, A.; Kwong, S.; Zhang, Q. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **2014**, *18*, 114–130.
9. Auer, P.; Cesa-Bianchi, N.; Fischer, P. Finite-time Analysis of the Multiarmed Bandit Problem. *Mach. Learn.* **2002**, *47*, 235–256.
10. Gonçalves, R.A.; Kuk, J.N.; Almeida, C.P.; Venske, S.M. Decomposition Based Multiobjective Hyper Heuristic with Differential Evolution. In *Computational Collective Intelligence*; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2015; pp. 129–138.
11. Gonçalves, R.A.; Almeida, C.P.; Pozo, A. Upper Confidence Bound (UCB) Algorithms for Adaptive Operator Selection in MOEA/D. In *Evolutionary Multi-Criterion Optimization*; Lecture Notes in Computer Science, Springer International Publishing: Cham, Switzerland, 2015; pp. 411–425.
12. Zhang, Q.; Zhou, A.; Zhao, S.; Suganthan, P.N.; Liu, W.; Tiwari, S. Multiobjective optimization test instances for the CEC 2009 special session and competition. University of Essex, Colchester, UK and Nanyang Technological University, Singapore, Special Session on Performance Assessment of Multi-Objective Optimization Algorithms. Technical Report. 2008; pp. 1–30.
13. Almeida, C.; Gonçalves, R.; Venske, S.; Lüders, R.; Delgado, M. Hyper-heuristics using multi-armed bandit models for multi-objective optimization. *Appl. Soft Comput.* **2020**, *95*, 106520.
14. Guizzo, G.; Fritsche, G.M.; Vergilio, S.R.; Pozo, A.T.R. A Hyper-Heuristic for the Multi-Objective Integration and Test Order Problem. In Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO'15), Madrid, Spain, 11–15 July 2015; ACM: New York, NY, USA, 2015; pp. 1343–1350.
15. ao, W.A.; Colanzi, T.E.; Vergilio, S.R.; Pozo, A. A Multi-objective Optimization Approach for the Integration and Test Order Problem. *Inf. Sci.* **2014**, *267*, 119–139.
16. Guizzo, G.; Vergilio, S.R.; Pozo, A.T.R. Evaluating a Multi-objective Hyper-Heuristic for the Integration and Test Order Problem. In Proceedings of the 2015 Brazilian Conference on Intelligent Systems (BRACIS), Natal, Brazil, 4–7 November 2015; SBC: Natal, Brazil, 2015; pp. 1–6.
17. de Carvalho, V.R. Uma Hiper-Heurística de Seleção Baseada em Decomposição Para Estabelecer Sequências de Módulos Para o Teste de Software. MsC Thesis, Universidade Federal do Paraná (UFPR), Curitiba, Brazil, 2015.
18. 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.
19. Liao, X.; Li, Q.; Yang, X.; Zhang, W.; Li, W. Multiobjective optimization for crash safety design of vehicles using stepwise regression model. *Struct. Multidiscip. Optim.* **2008**, *35*, 561–569.
20. 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.
21. Zitzler, E.; Laumanns, M.; Thiele, L. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*; EUROGEN 2001: Barcelona, Spain, 2001; pp. 95–100.
22. Fonseca, C.M.; Fleming, P.J. Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms. I. A Unified Formulation. *IEEE Trans. Syst. Man Cybern. Part A* **1998**, *28*, 26–37.
23. Vrugt, J.A.; Robinson, B.A. Improved evolutionary optimization from genetically adaptive multimethod search. *Proc. Natl. Acad. Sci. USA* **2007**, *104*, 708–711.
24. Li, W.; Özcan, E.; John, R. Multi-objective evolutionary algorithms and hyper-heuristics for wind farm layout optimisation. *Renew. Energy* **2017**, *105*, 473–482.
25. Zitzler, E.; Künzli, S. Indicator-Based Selection in Multiobjective Search. In *Parallel Problem Solving from Nature—PPSN VIII*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3242, pp. 832–842.
26. Tran, R.; Wu, J.; Denison, C.; Ackling, T.; Wagner, M.; Neumann, F. Fast and Effective Multi-objective Optimisation of Wind Turbine Placement. In Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO'13), Amsterdam, The Netherlands, 6–10 July 2013; ACM: New York, NY, USA, 2013; pp. 1381–1388.
27. Deb, K.; Thiele, L.; Laumanns, M.; Zitzler, E. Scalable Test Problems for Evolutionary Multiobjective Optimization. In *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*; Springer: London, UK, 2005; pp. 105–145.
28. de Carvalho, V.R.; Sichman, J.S. Applying Copeland Voting to Design an Agent-Based Hyper-Heuristic. In Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, São Paulo, Brazil, 8–12 May 2017; pp. 972–980.
29. de Carvalho, V.R.; Sichman, J.S. Multi-Agent Election-Based Hyper-Heuristics. In Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 13–19 July 2018; pp. 5779–5780.
30. Nurmi, H. *Voting Systems for Social Choice*; Springer: Dordrecht, The Netherlands, 2010; pp. 167–182.
31. Kukkonen, S.; Lampinen, J. GDE3: The third evolution step of generalized differential evolution. In Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2–5 September 2005; Volume 1; pp. 443–450.

32. de Carvalho, V.R.; Sichman, J.S. Solving real-world multi-objective engineering optimization problems with an Election-Based Hyper-Heuristic. In Proceedings of the OptMAS 2018: International Workshop on Optimisation in Multi-Agent Systems, AAMAS 2018, Stockholm, Sweden, 10–15 July 2018.
33. Tapabrata, R.; Kang, T.; Seow, K.C. Multiobjective Design Optimization by an Evolutionary Algorithm. *Eng. Optim.* **2001**, *33*, 399–424.
34. 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.
35. Ghiassi, M.; DeVor, R.; Dessouky, M.; Kijowski, B. An application of multiple criteria decision-making principles for planning machining operations. *IIE Trans.* **1984**, *16*, 106–114.
36. de Carvalho, V.R.; Larson, K.; Brandão, A.A.F.; Sichman, J.S. Applying Social Choice Theory to Solve Engineering Multi-objective Optimization Problems. *J. Control Autom. Electr. Syst.* **2020**, *31*, 119–128.
37. Hansen, M.P. Use of substitute scalarizing functions to guide a local search based heuristic: The case of moTSP. *J. Heuristics* **2000**, *6*, 419–431.
38. Institut Für Neuroinformatik. Black Box Optimization Competition, EMO'2017 Real-World Problems. 2017. Available online: https://www.ini.rub.de/PEOPLE/glasmtbl/projects/bbcomp/ (accessed on 10 April 2021).
39. Stadler, W.; Dauer, J. Multicriteria Optimization in Engineering: A Tutorial and Survey. *Struct. Optim. Status Promise* **1993**, *150*, 211–249.
40. Golinski, J. Optimal synthesis problems solved by means of nonlinear programming and random methods. *J. Mech.* **1970**, *5*, 287–309.
41. Quagliarella, D.; Vicini, A. Sub-population policies for a parallel multiobjective genetic algorithm with applications to wing design. In Proceedings of the SMC'98 Conference Proceedings, 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218), San Diego, CA, USA, 14 October 1998; Volume 4, pp. 3142–3147.
42. Poloni, C.; Mosetti, G.; Contessi, S. Multi objective optimization by GAs: Application to system and component design. In *ECCOMAS'96: Computational Methods in Applied Sciences'96*; John Wiley & Sons, Ltd.: Hoboken NJ, USA, 1996; pp. 1–7.
43. Li, W.; Özcan, E.; John, R.; Drake, J.H.; Neumann, A.; Wagner, M. A modified indicator-based evolutionary algorithm (mIBEA). In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, 5–8 June 2017; pp. 1047–1054.
44. Santiago, A.; Dorronsoro, B.; Fraire, H.J.; Ruiz, P. Micro-Genetic algorithm with fuzzy selection of operators for multi-Objective optimization: FAME. *Swarm Evol. Comput.* **2021**, *61*, 100818.
45. Karafotias, G.; Hoogendoorn, M.; Eiben, A.E. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Trans. Evol. Comput.* **2015**, *19*, 167–187.
46. Coello, C. *Evolutionary Algorithms for Solving Multi-Objective Problems*; Springer: New York, NY, USA, 2007.
47. Adra, S.F. Improving Convergence, Diversity and Pertinency in Multiobjective Optimisation. Ph.D. Thesis, Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, UK, 2007.
48. Atahran, A.; Lenté, C.; T'kindt, V. A Multicriteria Dial-a-Ride Problem with an Ecological Measure and Heterogeneous Vehicles. *J. Multi-Criteria Decis. Anal.* **2014**, *21*, 279–298.
49. Mariani, T.; Vergilio, S.R.; Colanzi, T.E. Optimizing Aspect-Oriented Product Line Architectures with Search-Based Algorithms. In *Search-Based Software Engineering*; Barros, M., Labiche, Y., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 173–187.
50. Hamdani, T.M.; Won, J.M.; Alimi, A.M.; Karray, F. Multi-objective feature selection with NSGA II. In Proceedings of the International Conference on Adaptive and Natural Computing Algorithms, Warsaw, Poland, 11–14 April 2007; pp. 240–247.
51. Goranova, M.; Contreras-Cruz, M.A.; Hoyle, A.; Ochoa, G. Optimising Antibiotic Treatments with Multi-objective Population-based Algorithms. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; pp. 1–7.
52. Dorigo, M.; Blum, C. Ant colony optimization theory: A survey. *Theor. Comput. Sci.* **2005**, *344*, 243–278.
53. Lalbakhsh, P.; Zaeri, B.; Lalbakhsh, A. An Improved Model of Ant Colony Optimization Using a Novel Pheromone Update Strategy. *IEICE Trans. Inf. Syst.* **2013**, *96*, 2309–2318.
54. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
55. Nebro, A.J.; Durillo, J.J.; Garcia-Nieto, J.; Coello Coello, C.A.; Luna, F.; Alba, E. SMPSO: A new PSO-based metaheuristic for multi-objective optimization. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making(MCDM), Nashville, TN, USA, 30 March–2 April 2009; pp. 66–73.
56. 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; Volume 3, pp. 1958–1962.
57. Marinakis, Y.; Marinaki, M. Combinatorial Expanding Neighborhood Topology Particle Swarm Optimization for the Vehicle Routing Problem with Stochastic Demands. In Proceedings of the 15th annual Conference on Genetic and Evolutionary Computation (GECCO'13), Amsterdam, The Netherlands, 6–10 July 2013; Association for Computing Machinery: New York, NY, USA, 2013; pp. 49–56.

58. de Carvalho, V.R.; Pozo, A.T.R. Um estudo sobre otimização por partículas aplicado ao problema de roteamento de veículos com demandas estocásticas. In Proceedings of the Encontro Nacional de Inteligência Artificial e Computacional (ENIAC), São Carlos, Brazil, 19–23 October 2014.
59. Lalbakhsh, A.; Afzal, M.U.; Esselle, K.P. Multiobjective Particle Swarm Optimization to Design a Time-Delay Equalizer Metasurface for an Electromagnetic Band-Gap Resonator Antenna. *IEEE Antennas Wirel. Propag. Lett.* **2017**, *16*, 912–915.
60. Lalbakhsh, A.; Afzal, M.U.; Esselle, K.P.; Smith, S. Design of an artificial magnetic conductor surface using an evolutionary algorithm. In Proceedings of the 2017 International Conference on Electromagnetics in Advanced Applications (ICEAA), Verona, Italy, 11–15 September 2017; pp. 885–887.
61. Lalbakhsh, A.; Afzal, M.U.; Zeb, B.A.; Esselle, K.P. Design of a dielectric phase-correcting structure for an EBG resonator antenna using particle swarm optimization. In Proceedings of the 2015 International Symposium on Antennas and Propagation (ISAP), Hobart, TAS, Australia, 9–12 November 2015; pp. 1–3.
62. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359.
63. Elarbi, M.; Bechikh, S.; Ben Said, L.; Datta, R. Multi-objective Optimization: Classical and Evolutionary Approaches. In *Recent Advances in Evolutionary Multi-objective Optimization*; Springer International Publishing: Cham, Switzerland, 2017; pp. 1–30.
64. Zitzler, E.; Thiele, L. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *Trans. Evol. Comput.* **1999**, *3*, 257–271.
65. Van Veldhuizen, D.A. Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. Ph.D. Thesis, Air Force Institute of Technology, Wright Patterson AFB, Dayton, OH, USA, 1999.
66. Goh, C.K.; Tan, K.C. A Competitive-Cooperative Coevolutionary Paradigm for Dynamic Multiobjective Optimization. *IEEE Trans. Evol. Comput.* **2009**, *13*, 103–127.
67. Tan, K.C.; Lee, T.; Khor, E. Evolutionary Algorithms for Multi-Objective Optimization: Performance Assessments and Comparisons. *Artif. Intell. Rev.* **2002**, *17*, 251–290.
68. Wooldridge, M. *An Introduction to Multiagent Systems*; John Wiley & Sons: Hoboken NJ, USA, 2009.
69. Copeland, A.H. *A Reasonable Social Welfare Function*; University of Michigan: Ann Arbor, MI, USA, 1951.
70. Bradstreet, L.; Barone, L.; While, L.; Huband, S.; Hingston, P. Use of the WFG Toolkit and PISA for Comparison of MOEAs. In Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making, Honolulu, HI, USA, 1–5 April 2007; pp. 382–389.
71. Nebro, A.J.; Durillo, J.J.; Vergne, M. Redesigning the jMetal Multi-Objective Optimization Framework. In *Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO Companion'15)*; ACM: New York, NY, USA, 2015; pp. 1093–1100.
72. Gunawan, S.; Azarm, S. Multi-objective robust optimization using a sensitivity region concept. *Struct. Multidiscip. Optim.* **2005**, *29*, 50–60.
73. Giotis, A.; Emmerich, M.; Naujoks, B.; Giannakoglou, K.; Bäck, T. Low-cost stochastic optimization for engineering applications. In *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems (EUROGEN-2001)*; Springer International Publishing: Cham, Switzerland, 2001; pp. 361–366.
74. Deb, K.; Sundar, J. Reference Point Based Multi-objective Optimization Using Evolutionary Algorithms. In Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO'06), Seattle, MA, USA, 8–12 July 2006; ACM: New York, NY, USA, 2006; pp. 635–642.
75. Yang, X.S. Multiobjective firefly algorithm for continuous optimization. *Eng. Comput.* **2013**, *29*, 175–184.
76. Zhang, Q.; Li, H. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Trans. Evol. Comput.* **2007**, *11*, 712–731.
77. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18.
78. Harman, M.; Jones, B.F. Search-based software engineering. *Inf. Softw. Technol.* **2001**, *43*, 833–839.
79. Jamshidi, M.B.; Lalbakhsh, A.; Talla, J.; Peroutka, Z.; Hadjilooei, F.; Lalbakhsh, P.; Jamshidi, M.; Spada, L.L.; Mirmozafari, M.; Dehghani, M.; et al. Artificial Intelligence and COVID-19: Deep Learning Approaches for Diagnosis and Treatment. *IEEE Access* **2020**, *8*, 109581–109595.
80. Jamshidi, M.B.; Lalbakhsh, A.; Talla, J.; Peroutka, Z.; Roshani, S.; Matousek, V.; Roshani, S.; Mirmozafari, M.; Malek, Z.; La Spada, L.; et al. Deep Learning Techniques and COVID-19 Drug Discovery: Fundamentals, State-of-the-Art and Future Directions. *Emerg. Technol. Dur. Era COVID-19 Pandemic* **2021**, *348*, 9.