


Article

Multi-Objective Majority–Minority Cellular Automata Algorithm for Global and Engineering Design Optimization

Juan Carlos Seck-Tuoh-Mora ^{*,†} , Ulises Hernandez-Hurtado [†] , Joselito Medina-Marín ,
Norberto Hernández-Romero  and Liliana Lizárraga-Mendiola 

Academic Area of Engineering and Architecture, Institute of Basic Sciences and Engineering,
Autonomous University of the State of Hidalgo, Pachuca 42184, Hidalgo, Mexico;
ulixexido@comunidad.unam.mx (U.H.-H.); jmedina@uaeh.edu.mx (J.M.-M.);
nhromero@uaeh.edu.mx (N.H.-R.); mendiola@uaeh.edu.mx (L.L.-M.)

* Correspondence: jseck@uaeh.edu.mx

[†] These authors contributed equally to this work.

Abstract: When dealing with complex models in real situations, many optimization problems require the use of more than one objective function to adequately represent the relevant characteristics of the system under consideration. Multi-objective optimization algorithms that can deal with several objective functions are necessary in order to obtain reasonable results within an adequate processing time. This paper presents the multi-objective version of a recent metaheuristic algorithm that optimizes a single objective function, known as the Majority–minority Cellular Automata Algorithm (MmCAA), inspired by cellular automata operations. The algorithm presented here is known as the Multi-objective Majority–minority Cellular Automata Algorithm (MOMmCAA). The MOMmCAA adds repository management and multi-objective search space density control to complement the performance of the MmCAA and make it capable of optimizing multi-objective problems. To evaluate the performance of the MOMmCAA, results on benchmark test sets (DTLZ, quadratic, and CEC-2020) and real-world engineering design problems were compared against other multi-objective algorithms recognized for their performance (MOLAPO, GS, MOPSO, NSGA-II, and MNMA). The results obtained in this work show that the MOMmCA achieves comparable performance with the other metaheuristic methods, demonstrating its competitiveness for use in multi-objective problems. The MOMmCAA was implemented in MATLAB and its source code can be consulted in GitHub.

Keywords: majority–minority cellular automata algorithm (MmCAA); multi-objective optimization; metaheuristic; cellular automata; real-world engineering problems



Citation: Seck-Tuoh-Mora, J.C.; Hernandez-Hurtado, U.; Medina-Marín, J.; Hernández-Romero, N.; Lizárraga-Mendiola, L. Multi-Objective Majority–Minority Cellular Automata Algorithm for Global and Engineering Design Optimization. *Algorithms* **2024**, *17*, 433. <https://doi.org/10.3390/a17100433>

Academic Editors: Łukasz Knypiński, Ramesh Devarapalli and Marcin Kaminski

Received: 28 August 2024

Revised: 23 September 2024

Accepted: 24 September 2024

Published: 27 September 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Many real-world problems in engineering and other research areas require multi-objective optimization, where it is necessary to find a set of solutions in the search space that are well-distributed along the Pareto-optimal front. Generally, in this type of problem the computation of possible solutions must consider the existence of two or more conflicting objective functions. A multi-objective optimization problem can be defined as follows:

$$\min h(z) = (h_1(z), h_2(z), \dots, h_m(z)) \quad (1)$$

where $z \in Q, m \geq 2$

where each $h_i(z)$ is a real-valued scalar function, $h(z)$ is the set of objective or cost functions that produce an m -dimensional vector in the \mathbb{R}^m -objective space when evaluated, $z = (z_1, z_2, \dots, z_n)$ is an n -dimensional vector in the search space \mathbb{R}^n , and $Q \subseteq \mathbb{R}^n$ is the set of all feasible solutions of Equation (1).

In this type of problem, identifying the set of best possible solutions can in many cases be highly complicated or impossible. Rather than looking for globally optimal solutions to a

multi-objective problem, it is possible to instead seek to compute satisfactory solutions that can be obtained in adequate time, mainly to find the Pareto optimal (PS) set of solutions. The mapping from the PS to the objective space is the Pareto front (*PF*). Examples of classical optimization techniques adapted to multi-objective problems are the Weighted Sum Method [1], ϵ -constraint method [2], goal programming [3], and lexicographic ordering [4] among others.

Multi-objective optimization evolutionary algorithms (MOEAs) are highly suitable for solving problems involving multiple objectives because they can generate a set of *PF*-approximate solutions in a single run [5]. In recent decades, many multi-objective optimization algorithms have been derived from classical single-objective metaheuristics, showing efficiency and effectiveness on various complex problems. Single-objective metaheuristics are optimization techniques that focus on finding a single optimal solution. Common examples include genetic algorithms (GA) [6], particle swarm optimization (PSO) [7], ant colony optimization (ACO) [8], and simulated annealing (SA) [9].

Several techniques based on single-objective metaheuristics have been developed to address multi-objective problems. Examples of the most outstanding options include the Non-Dominated Sorting Genetic Algorithm (NSGA-II), which extends GAs to handle multiple objectives using a non-dominated sorting scheme and population diversity [10]; the Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D), which splits a multi-objective problem into several single-objective problems [11]; SPEA2, which uses a list of non-dominated solutions and assigns strengths to each solution to guide the search process [12]; Multi-objective PSO (MOPSO), which is an extension of PSO that allows it to handle multiple objectives while maintaining an archive of non-dominated solutions [13]; Multi-Objective Simulated Annealing (MOSA), which extends simulated annealing to handle multiple targets in order to maintain a balance between exploration and exploitation [14]; the Multi-objective Ant Lion Optimizer (MOALO), which expands the Ant Lion Optimizer by applying a repository to store non-dominated solutions in the Pareto set [15]; the Multi-objective Multi-Verse Optimizer (MOMVO), which builds on the MVO to compare and optimize test and practical problems [16]; and the Multi-objective ACO (MOACO), which adapts the ACO for multi-objective optimization using multiple populations to improve computational efficiency [17,18]. MOACO has also been employed to address multi-objective problems in airline crew turnover [19] and to improve the supply chain configurations [20].

Another work that is an extension of a recent metaheuristic algorithm is the Multi-objective Salp Swarm Algorithm (MSSA) [21]. Inspired by the swarming behavior of sea salps, the MSSA splits the population into a leader and followers, using an external file to store the non-dominated solutions. This approach shows adequate convergence and coverage of the PS. The interplay of several PSO algorithms for simultaneous optimization of single objectives in a multi-objective problem (MPMO) is described in [22] using multiple populations. In discrete problems, the MPMOGA is a new algorithm inspired by the GA which uses multiple populations to solve problems with multiple objectives. It was used in [23] to address the job-shop scheduling problems, obtaining satisfactory results. Multi-Objective Heat Transfer Search (MOHTS) based on Heat Transfer Search was proposed in [24] to optimize structural multi-objective problems, obtaining better results compared with other algorithms based on ant colonies and symbolic organisms. In related research, a multi-objective version of the symbolic organism algorithm was presented in [25] for optimal reinforcement design.

Multi-Objective Teaching–Learning-based Optimization (MOTLBO) is an extension of the Teaching–Learning-based Optimization (TLBO) algorithm. This approach uses two main phases: the teacher phase, where solutions are improved based on the best individual, and the learner phase, where solutions are optimized through knowledge sharing between individuals [26]. Multi-Objective Thermal Exchange Optimization (MOTEO), inspired by the principles of thermodynamics and heat exchange, seeks to solve optimization problems by considering multiple criteria simultaneously [27]. Multi-objective Plasma Generation

Optimization (MOPGO) is inspired by the generation and behavior of plasmas, where charged particles interact and move towards lower energy states [28]. The Multi-Objective Crystal Structure Algorithm (MOCSA) is motivated by the formation and organization of crystalline structures; solutions resemble atoms that organize themselves into configurations that minimize the energy of the system [29]. The Multi-Objective Forest Optimization Algorithm (MOFOA) follows the dynamics and ecology of forests; solutions resemble trees competing for resources, allowing the solutions (trees) to evolve and adapt in a competitive and cooperative environment [30]. The Competitive Mechanism Integrated Multi-Objective Whale Optimization Algorithm with Differential Evolution (CMI-MOWOA-DE) combines whale social behavior and differential evolution; the solutions simulate whale movement and hunting strategy, while differential evolution introduces variation and diversification to achieve a balance between multiple conflicting objectives. The Multi-Objective Harris Hawks Optimizer (MOHHO) is an extension of the Harris Hawks Optimizer (HHO) algorithm, which evokes the cooperative hunting strategies of Harris Hawks [31]. The Marine Predators Algorithm (MPA) emulates the behavior of species such as sharks and dolphins for multi-objective optimization, using search tactics and solution space exploitation to optimize multiple objectives [32]. The Multi-Objective Sine–Cosine Algorithm (MOSCA) is a variant of the Sine–Cosine Algorithm (SCA) that uses sine and cosine functions to guide the exploration and exploitation of the search space, dynamically adjusting the positions of candidate solutions to maintain diversity and ensure convergence to the FP [33]. The Multi-objective Atomic Orbital Search (MAOS) algorithm is based on the concept of atomic orbitals from quantum chemistry; the potential solutions are treated as electrons in different orbitals, and the search process resembles the movement of these electrons to reach lower energy configurations [34]. In the branch-and-bound framework for continuous global multi-objective optimization, the search space is recursively divided into smaller subregions, then lower and upper bounds are computed for the objective functions in these subregions. Subregions that cannot contain optimal solutions are discarded, which reduces the overall search space. Multi-Objective Differential Evolution (MODE) uses a population of candidate solutions that evolve through mutation operators. The multi-objective optimization method based on adaptive parameter harmony search algorithm simulates the improvisation process of musicians searching for the best harmony, dynamically adapting its parameters using memory and tuning operators to explore new solutions and preserve the best ones [35]. The Guided Population Archive Whale Optimization Algorithm (GPA-WOA) is a variant of the Whale Optimization Algorithm (WOA) that simulates the hunting behavior of humpback whales and uses guides or benchmark solutions to direct the search and improve convergence to the FP; a population file is dynamically updated to preserve diversity and ensure that solutions are optimal and well-distributed [36]. The quantum-inspired Decomposition-based Quantum Salp Swarm Algorithm (DQSSA) combines quantum mechanical principles with the swarming behavior of salps to divide multi-objective problems into more tractable subproblems, allowing a set of well-distributed optimal solutions to be found at the FP [37].

The above works are just a sample of the many single-objective algorithms that have recently been extended in various ways to deal with multi-objective problems. Single-objective optimization algorithms inspired by cellular automata are practical and have competitive results on these types of problems compared to more recent metaheuristics.

For instance, Cellular Particle Swarm Optimization (CPSO) is a variant of the classical PSO algorithm that organizes particles into a cellular lattice structure in which each particle only interacts with its nearest neighbors, thereby improving exploration and reducing the probability of premature convergence [38]. Island Cellular Model Differential Evolution combines the principles of Differential Evolution (DE) with a distributed population structure; a cellular scheme divides the population into subpopulations, which promotes genetic diversity, reduces premature convergence, and enhances exploration capability [39]. The Continuous-State Cellular Automata Algorithm (CCAA) is inspired by cellular automata but adapted to work with continuous rather than discrete variables. In this algorithm,

individuals (or smart-cells) are organized in a spatial grid; each cell updates its state (candidate solution) based on the solutions of its local neighbors. Continuous states allow for finer exploration of the search space, while the restricted neighborhood structure favors a balance between local exploitation and global exploration [40]. The Cellular Learning Automata and Reinforcement Learning (CLARL) approach combines the principles of learning automata and reinforcement learning. In this method, learning automata are organized in a cellular mesh, where each automaton represents a potential solution and adapts its behavior through local interactions and reward-based feedback. This scheme allows for learning strategies that improve the system's dynamic adaptability [41]. The Reversible Elementary Cellular Automata Algorithm (RECAA) uses reversible rules, meaning that the system can return to previous states without losing information. In this algorithm, each potential solution follows simple local rules to update its state but with the property of reversibility, enabling a more controlled and efficient search space exploration [42].

However, only a few works have applied the concept of cellular automata for general multi-objective optimization. One of the most representative examples is the Cellular Ant Algorithm (CAA) for multi-objective optimization, which combines the ant colony structure with a cellular mesh in which ants only interact with their close neighbors. This mechanism simultaneously optimizes several objective functions, achieving balanced solutions to complex problems with multiple criteria [43]. Multi-objective Cellular Automata Optimization is another approach that applies cellular automata. Potential solutions are cells in a network that evolve based on local rules and interaction with their neighbors. This approach seeks to reach a balance by facilitating the identification of solutions [44]. Cellular Multi-objective Particle Swarm Optimization (CMPSO) is a variant of PSO in which particles are arranged in a cellular structure and only interact with their close neighbors. This promotes solution diversity by limiting global influences and encourages better exploration, and it is beneficial in applications that require simultaneous optimization of several criteria [45]. Cellular Teaching–Learning–Based Optimization (CTLBO) is a teaching–learning-based approach to optimization. In this method, solutions are organized in a cellular structure, where each cell represents an individual who learns from its neighbors and a virtual teacher who guides the process. This approach enhances the algorithm's ability to adapt to dynamic changes for multiple objectives that may vary over time [46].

Following this trend, a recent single-objective optimization algorithm is the Majority–minority Cellular Automata Algorithm (MmCAA), which was tested on several test problems in multiple dimensions and for various applications in engineering, obtaining satisfactory results against other well-recognized algorithms [47].

This paper presents a multi-objective version of this algorithm called MOMmCAA. This algorithm is inspired by the local behavior of cellular automata, particularly the majority and minority rules, which are intermixed and able to generate complex behaviors in order to perform the tasks of exploration and exploitation in the search space.

The problem to be addressed in this work is the optimization of multi-objective problems using a modification of the MmCAA to obtain an adequate approximation of its PS. Although multi-objective algorithms are continuously proposed in the specialized literature, they have yet to fully exploit the advantages offered by the different cellular automata rules, such as the diversities and richness of their dynamic behaviors and their easy implementation. Thus, this work aims to test and demonstrate the feasibility of modifying the MmCAA to deal with multi-objective problems in a manner comparable to current well-recognized algorithms for performing this task. The manuscript's originality lies in the fact that it is the first to propose an algorithm for multi-objective optimization inspired by cellular automata using majority and minority rules, and is complemented by managing a repository to control the density of solutions in the FP.

To test the performance of the MOMmCAA, we used the DLTZ benchmark, ten quadratic problems, and ten CEC2020 problems. The proposed algorithm was also tested on two practical engineering problems, obtaining satisfactory results. In these cases, five other algorithms were also considered for comparison: Multi-Objective Lightning Attachment Procedure Optimization (MOLAPO) [48], Grid Search (GS) [49], Multi-Objective Particle Swarm Optimization (MOPSO) [13], the Non-dominated Sorting Genetic Algorithm (NSGA-II) [10], and the Multi-objective Nelder–Mead Algorithm (MNMA) [50].

Non-parametric Wilcoxon statistical tests were performed to show the statistical significance of the experiments. The results indicate that the proposed algorithm ranks among the best with respect to the other methods used in this work.

The rest of this article is organized as follows. Section 2 presents the details of the Multi-Objective Majority–minority Cellular Automata Algorithm (MOMmCAA); Section 3 presents the results of our experiments on various test benches (DTLZ benchmark, ten quadratic problems, and ten CEC2020 problems), providing a statistical comparison of the MOMmCAA through the Wilcoxon test that relates it to other multi-objective algorithms recognized for their performance; Section 4 describes the application of the MOMmCAA to two practical engineering problems (design of a four-bar truss and a disk brake); finally, Section 5 provides the paper’s conclusions.

2. The Proposed Multi-Objective Majority–Minority Cellular Automata Algorithm (MOMmCAA)

This section briefly explains the concept of cellular automata, the general characteristics of the Majority–minority Cellular Automata Algorithm, and the multi-objective implementation of this algorithm. The concept of hypercubes is used to delimit a repository for managing the PS solutions generated by the algorithm.

2.1. Basic Concepts of Cellular Automata with Majority Rule

Cellular Automata (CA) are dynamic systems of cells that initially take a value from a finite set of possible states. The dynamics of CA proceed in discrete steps, making CA discrete systems in time and space. At each step, a cell considers its current state and that of its close neighbors in order to update its state at the next time step. In this way, a mapping from blocks of states to individual states is called an evolution rule. CA can generate chaotic and complex global behaviors depending on the evolution rule that defines their local mapping. Because of this, they have been widely investigated and applied in various engineering and computational problems [51,52].

One of the rules of evolution extensively studied in recent work is the majority rule. In this rule, each cell takes its new state as the most common state in its local neighborhood. The dynamics of this rule are characterized by patchy patterns that stabilize as the system’s evolution progresses. Its counterpart is the minority rule, which takes the least common element of each neighborhood to update the state of a cell in the next generation. The evolution of the minority rule is characterized by the fact that it does not tend quickly to a fixed or periodic state, as a minority state tends to become the majority and vice versa, resulting in oscillating global dynamics.

Figure 1 shows various dynamic behaviors of the majority rule and minority rule along with the application of the majority rule with probability in cellular automata of two states and various neighborhood sizes. In these examples, 500 cells and 250 evolutions were used. Evolution generates a periodic pattern for the original majority rule, while the minority rule generates a chaotic pattern of heterogeneous triangular shapes. When the rules are alternated probabilistically, the result is the formation of complex patterns in which non-periodic structures are combined with a stable background.



Figure 1. Examples of cellular automata with two states and a neighborhood size of 3, applying majority, minority, and majority with probability evolution rules.

This combination of majority and minority rules was used as inspiration to define the Majority–minority Cellular Automata Algorithm (MmCAA) for single-objective optimization [47]. The inspiration behind the MmCAA is to emulate the dynamic behavior of applying majority and minority rules in cellular automata.

The MmCAA starts by generating a random population S of n_S smart-cells, where each smart-cell is represented as $s \in \mathbb{R}^n$. The dynamics of each s is defined by a set of rules, with one of them chosen randomly to improve the position of the smart-cell. The rules take information from other smart-cells to generate new neighbors, from which the best one is selected to upgrade the smart-cell position. With this mechanism, the positions of all smart-cells in the population are improved and the system evolves iteratively during the optimization process. The rules used by the MmCAA for smart-cell evolution are as follows.

The majority (minority) rule applied to a single smart-cell is described in Algorithm 1. The input is a smart-cell s_i for $1 \leq i \leq n_S$ and a weight parameter $prop_w$ that defines a limit on the change in the values of s_i . The rule takes the differences cm between the values in s_i and the most repeated element el in the smart-cell, and $rand$ generates a random value between 0 and 1. A new solution $evol$ is formed by taking the differences between the original smart-cell and cm randomly weighted between 0 and $prop_w$. This rule helps to bring the values of s_i closer to the most repeated value el .

Algorithm 1: Majority (minority) rule for a single smart-cell [47]

Result: New smart-cell $evol$

Input: s_i , $prop_w$;

el = most repeated element in s_i ;

forall k in $length(s_i)$ **do**

$cm(k) = s_i(k) - el$;

end

$cm = cm * prop_w * rand$;

$evol = s_i - cm$;

The majority (minority) rule applied to a neighboring smart-cell is described in Algorithm 2. The most repeated value of a neighboring smart-cell s_j is taken as the change factor, depending on the average weight of the neighbor cost $\mu(h(s_j))$. If the cost $\mu(h(s_i))$ is larger than $\mu(h(s_j))$, then the weight pon is large and there is a higher probability of changing s_i by taking a random ratio between $-prop_w$ and $prop_w$ of the most repeated element in s_j and modifying each randomly selected position in s_i .

Algorithm 2: Majority (minority) rule with one neighbor [47]

Result: New smart-cell *evol*
Input: $s_i, \mu(h(s_i)), s_j, \mu(h(s_j))$ *prop_w*;
evol = s_i ;
 $sm = \mu(h(s_i)) + \mu(h(s_j))$;
 $pon = 1 - (\mu(h(s_j)) / sm)$;
 el = most repeated element in neighbor s_j ;
 $r = (rand * prop_w) - (prop_w / 2)$;
forall k in $length(evol)$ **do**
 if $rand \leq pon$ **then**
 $evol(k) = evol(k) + (r * el)$;
 end
end

The rule for rounding values in a smart-cell (Algorithm 3) consists of rounding off n_r to the most significant decimal values of the selected elements in s_i [42]. The least significant decimal digit is the n_r -th digit to the right of the decimal point. The least significant digit remains unchanged if the first non-significant digit is less than 5; otherwise, the least significant digit is incremented by 1. This rule is applicable to find proper parameters for optimization problems.

Algorithm 3: Rounding rule [42]

Result: New smart-cell *evol*
Input: $s_i, f(s_i), f(b_s), n_r$;
evol = s_i ;
 $sum = f(s_i) + f(b_s)$;
 $pon = 1 - (f(s_i) / sum)$;
forall k in $length(evol)$ **do**
 if $rand \leq pon$ **then**
 $evol(k) = round(evol(k), n_r)$;
 end
end

The adaptation of the majority and minority rules considers the elements of each solution that are repeated to a greater or lesser degree in the same smart-cell or in one or two additional smart-cells to obtain a new position. The randomness in choosing evolution rules and neighbors allows for access to the information in the rest of the population, thereby generating large and small changes in the position of a smart-cell, which favors the exploration and exploitation phases to escape from local optima and avoid the stagnation of the solutions. The MmCAA is presented in Figure 2. In (A), each smart-cell checks its neighbors using different majority and minority rules. These rules produce new solutions (B) and the best solution in the neighborhood is selected to update the smart-cell (C). The pseudo-code of MmCAA is described in (D).

2.2. Multi-Objective Majority–Minority Cellular Automata Algorithm (MOMmCAA)

The MmCAA was devised to solve single-objective optimization problems; therefore, it needs to be modified to deal with multi-objective problems. This results in the creation of the proposed multi-objective variant, MOMmCAA.

Taking as a basis the MmCAA inspired by the combination of majority and minority cellular automata and the handling of a solution repository on the Pareto front of the MOPSO algorithm, the MOMmCAA uses the following mechanisms for multi-objective optimization.

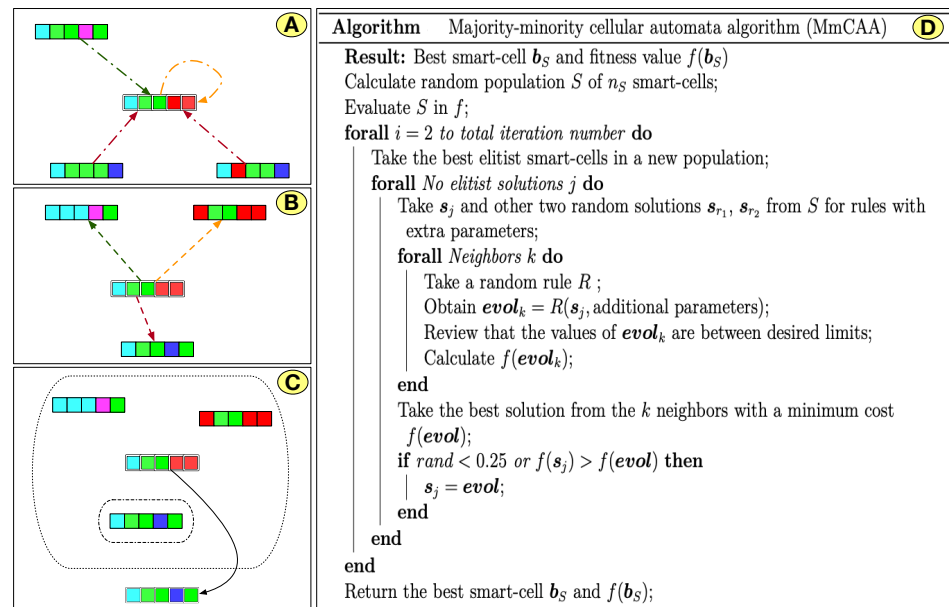


Figure 2. Majority-minority Cellular Automata Algorithm (MmCAA) [42].

Updating of each smart-cell: Each smart-cell solution generates a new set of neighboring solutions by taking its information or the information from one or two neighboring smart-cells (depending on the evolution rule that is randomly selected). Certain rules favor exploration by taking information from other smart-cells, while others favor exploitation by only taking information contained in the same smart-cell. From this set of neighbors, the one that is not dominated by the rest is used to update the smart-cell if it dominates it.

Repository with non-dominated solutions: The non-dominated smart-cells are stored in a repository, which also serves as a file to take neighbors when applying the different evolution rules that define the MOMmCAA optimization process. In order for a smart-cell to enter the repository, either it must dominate another solution or it must be the case that no other solution in the repository dominates it. The repository has a limited capacity; if a new smart-cell enters the repository, then the smart-cell that is dominated or the one that is in a region of the objective space with high density is deleted.

Hypercube density management in the objective space: Taking inspiration from the MOPSO mechanism, solutions in the PS are ranked depending on the density of the hypercube in which they are found in the solution space. If any other solution in the repository does not dominate a new solution and in turn is in a hypercube of lower density, then a solution that is in the hypercube with higher density is removed from the repository. This allows the repository to contain a better diversity of solutions. If a new solution is found in a new hypercube, then the boundaries of the solution space are expanded and the hypercube densities are recalculated. Figure 3 shows the handling of smart-cell selection in the repository using hypercubes; in (A), a new smart-cell replaces another smart-cell in a hypercube if it dominates it. In (B), if the new smart-cell falls into a higher-density hypercube and the repository is complete, then one of the smart-cells is randomly removed. In (C), if the new smart-cell falls into a less dense hypercube and the repository is complete, then one smart-cell is randomly removed from the denser hypercube. In (D), the hypercube boundaries are updated if the new smart-cell falls outside the current hypercube's boundaries.

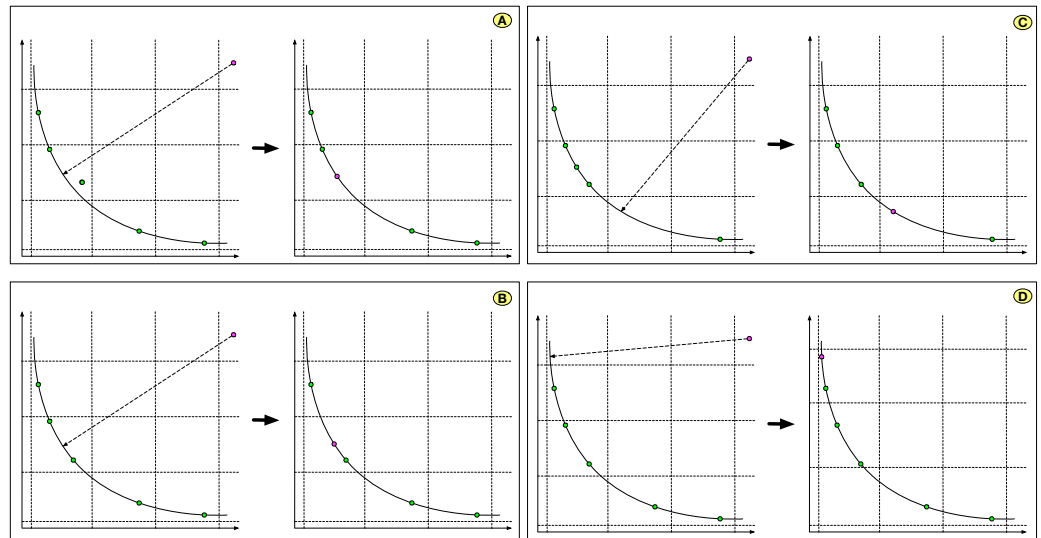


Figure 3. PS repository management using *PF* hypercubes.

The pseudocode of the MOMmCAA is presented in Algorithm 4.

Algorithm 4: Multi-Objective Majority–minority Cellular Automata Algorithm (MOMmCAA)

Result: Repository B_S of best smart-cells approximating *PS*

Generate random population S of n_S smart-cells;

Evaluate S in h ;

Initialize repository B_S of non-dominated smart-cells;

Calculate hypercube with n_{int} intervals and capacity cap ;

forall $i = 2$ to n_{it} **do**

 Keep the best n_{el} smart-cells in a new population;

forall $j = n_{el} + 1$ to n_S **do**

 Take s_j and other two random smart-cells s_{r_1}, s_{r_2} from S for rules requiring extra solutions;

forall $k = 1$ to n_{ne} **do**

 Choose a random rule R ;

 Obtain $evol_k = R(s_j, \text{additional parameters of the rule})$;

 Check that the n_d values of $evol_k$ are between lb and ub and correct if necessary;

 Calculate $h(evol_k)$;

end

 Choose the neighbor $evol$ that dominates s_j and is in a hypercube with equal or lower density than s_j from the k generated neighbors. In other case, conserve s_j ;

 If s_j has been improved, update B_S ;

 If B_S has been improved, update the hypercube;

end

end

Return the repository B_S with approximated *PS*;

2.3. Computational Complexity of the Proposed Algorithm

The computational time and space complexity of the MOMmCAA depends on the number of smart-cells N , number of objective functions to be evaluated m , management of the repository (where cap non-dominated solutions are stored), and total number of iterations n_{it} .

- **Smart-cell evaluation:** Each iteration evaluates all smart-cells, with a cost of $O(Nm)$.
- **Repository management:** The repository stores the non-dominated solutions. The cost of the Pareto dominance ordering mechanism can be high because the ordering of the non-dominated solutions has a complexity of $O(cap^2)$. Methods with similar

strategies attempt to reduce this cost by limiting the repository size; however, it is still costly, especially as the number of non-dominated solutions grows.

- **Iterations of the algorithm:** The total number of iterations impacts the complexity, as evaluations and repository management are performed at each step.

The overall complexity of the MOMmCAA can be approximated as $O(n_{it}(Nm + cap^2))$. In summary, the time complexity of the MOMmCAA is quadratic in terms of the repository size of non-dominated solutions. The space complexity is linear with respect to the population of smart-cells, the number of objectives, and the number of iterations. One of the advantages of using adaptive hypercubes is that the computational cost is better than when using a niche strategy such as the one used by NSGA-II [53]. The only case where both strategies have the same quadratic complexity is when the hypercubes are updated at each generation [54]. Thus, the complexity of the MOMmCAA is similar to that of MOPSO.

3. Computational Experiments Comparing MOMmCAA to Other Algorithms

MOLAPO, GS, MOPSO, NSGA-II, MOMVO, and MNMA were compared to MOMmCAA in order to identify the best performance in calculating Pareto-optimal solutions. The initial parameters of all described algorithms are summarized in Table 1. Each experiment employed 50 PS solutions and a maximum of 1000 iterations. The proposed algorithm was tested in 29 diverse case studies, including 27 unconstrained and constrained mathematical problems and two real-world engineering design problems.

Table 1. Parameters of the algorithms used for comparison: MOLAPO, GS, MOPSO, NSGA-II, MNMA, and MOMmCAA.

Algorithm	Parameters
MOLAPO	$FE = 2, cap = 50$
GS	$numparts = 20, cap = 50$
MOPSO	$C1 = C2 = 2, w = wmax - t \times (wmax - wmin) / (tmax),$ $wmax = 0.9, wmin = 0.4, cap = 50$
NSGA-II	$pcross = 0.7, ncross = 2 * round(pcross * 100 / 2), pmut = 0.4,$ $nmut = round(pmut * 100), \mu = 0.02, Sigma = 0.1 * (vmax - vmin)$
MNMA	$\delta^e = 2, \delta^{oc} = 0.5, \delta^{ic} = -0.5, \gamma = 5$
MOMmCAA	$cap = 50, n_s = 11, n_{ne} = 3, n_{int} = 5, prop_w = 5, 2 \leq n_r \leq 5$

The original Matlab implementations of these algorithms were taken directly from the web addresses indicated in the reference articles. The Matlab code of the MOMmCAA can be downloaded from Github using the link <https://github.com/juanseck/MOMmCAA> (accessed on 2 September 2024). The MOMmCAA and other algorithms were executed in Matlab 2015a on a PC with a 3.1 GHz Intel Xeon CPU with 64 GB of RAM using the macOS Sonoma operating system. Thirty independent runs were made for each algorithm on every benchmark function. A number of different metrics were used to compare the results of the algorithms, as described below.

Hypervolume (HV): The diverseness in the search space through the hypervolume metric was first introduced by Ulrich et al. to escalate the diversity in both decision space and objective space [55]. The HV of a set of solutions measures the size of the portion of the objective space dominated by those solutions as a group. In general, HV is favored because it captures both the closeness of the solutions to the optimal set and (to some extent) the distribution of solutions across the objective space in a single scalar. The HV value measures both convergence and diversity, and can be calculated using the equation

$$HV = \cup_s Z(s) \mid s \in PF, \quad (2)$$

where $Z(s)$ refers to the hypercube bounded by a solution s in the obtained PF. A larger HV value indicates a better approximation of the PF.

Contribution (C): The Contribution metric counts the number of PS points used in the combined solution of all algorithms. This metric is an extension of the Purity metric [56]

For two approximation Pareto sets A and B , where $B \subset A$, the C metric assigns A a higher measure than B .

For $O \geq 2$ MOEAs applied to a problem, let R_i be the non-dominated solutions obtained by the i -th MOEA for $1 \leq i \leq O$. The union of all these sets is $R = \cup_{i=1}^O R_i$. The set R^* of non-dominated solutions is calculated from R . Let r_i^* be the number of non-dominated sets in R^* obtained by the i -th MOEA:

$$r_i^* = \{s \mid s \in R_i \text{ and } s \in R^*\}. \quad (3)$$

Thus, the C_i metric of the i -th MOEA is defined as

$$C_i = \frac{|r_i^*|}{|R_i|}. \quad (4)$$

The C_i value may lie between $[0, 1]$, with a value nearer to 1 indicating better performance.

Epsilon Indicator (EI):

The Epsilon Indicator was defined in [57]. It measures the minimum value of the scalar ϵ required to make the Pareto front (PF) dominated by the approximation set S . Epsilon values fall within the range of $[1, \infty)$.

$$I_\epsilon(PF, S) = \inf_{\epsilon} \{ \epsilon \mid \forall s \in S, \exists b \in PF \text{ such that } b \leq \epsilon a \} \quad (5)$$

In this case, the output of the epsilon indicator function is $1/\epsilon$; a value in the $(0, 1]$ range with a value near 1 is a close fit with the solution set.

3.1. Benchmark Instances

A total of 27 benchmark instances with complicated characteristics were used to compare the performance of the proposed MOMmCAA: DLTZ1-DLTZ7, ten quadratic problems, and ten CEC2020 test instances. These problems exhibit various characteristics, such as a convex, concave, mixed, disconnected, or degenerated PFs and a multimodal, biased, deceptive, and nonlinear variable PS.

For each instance, the compared algorithms are ranked according to the performance metrics, with the ranks shown in square brackets. The mean rank (MR) for each algorithm for each instance is also presented in the tables. As a result of the Wilcoxon rank sum test at a 5% significance level, a result labeled + denotes that the compared algorithm outperforms the MOMmCAA; in contrast, − means that the MOMmCAA has a better performance than the compared algorithm, while \approx means that there is no statistically significant difference between MOMmCAA and the compared algorithm. The data in orange in every table show the best mean metric values yielded by the algorithms for each instance over 30 independent runs.

3.2. DLTZ Instances

Tables 2–4 present the results of the metric values obtained by algorithms.

As shown in Table 2, the MOMmCAA obtains significantly better HV values than MOLAPO, GS, MOPSO, NSGA-II, and MNMA for four, four, one, five, and seven out of the seven instances, respectively. Regarding the overall mean rankings, the MOMmCAA obtains the second optimal mean rank value, below MOPSO, followed by GS, NSGA-II, MOLAPO, and MNMA. The MOMmCAA has poor performance on the DLTZ1 and DLTZ3 test instances. In summary, the MOMmCAA is superior to the other four MOEAs on this metric. Table 3 shows that the MOMmCAA achieves seven, seven, six, six, and seven better C metric values than MOLAPO, GS, MOPSO, NSGA-II, and MNMA, respectively. This indicates the quality of the solutions obtained by the MOMmCAA. Table 4 summarizes the overall performance of six algorithms in terms of EI metric values. The MOMmCAA yields significantly better EI values than MOLAPO, GS, MOPSO, NSGA-II, and MNMA for six, four, one, seven, and five out of the seven instances, respectively. Overall, the EI statistics are similar to those for HV . Figure 4 plots the representative PFs obtained by the six comparison MOEAs. In summary, the MOMmCAA shows competitive performance on the DLTZ benchmark.

Table 2. Statistics (mean (std. dev.)) of Hypervolume metric values of the final populations obtained by MOLAPO, GS, MOPSO, NSGA-II, MNMA, and MOMmCAA over 30 independent runs on the DLTZ benchmark. The ranking of each algorithm is in braces.

Fn	MOLAPO	GS	MOPSO	NSGA-II	MNMA	MOMmCAA
DLTZ1	0.9942 (0.0019) [3]+	0.9999 (0.0035) [1]+	0.9962 (0.0157) [2]+	0.8081 (0.0372) [5]−	0.9579 (0.0121) [6]−	0.9867 (0.0028) [4]
DLTZ2	0.9960 (0.0039) [4]≈	0.9709 (0.0026) [6]−	0.9985 (0.0016) [1]+	0.9969 (0.0030) [2]≈	0.9850 (0.0057) [5]−	0.9969 (0.0033) [3]
DLTZ3	0.9905 (0.0045) [3]+	0.9989 (0.0022) [1]+	0.9973 (0.0032) [2]+	0.6326 (0.0529) [6]−	0.7055 (0.0608) [5]−	0.9746 (0.0058) [4]
DLTZ4	0.8115 (0.0480) [3]−	0.1597 (0.0039) [6]−	0.9916 (0.0346) [1]+	0.6762 (0.1710) [4]−	0.6434 (0.1716) [5]−	0.8940 (0.0588) [2]
DLTZ5	0.0733 (0.0260) [6]−	0.3479 (0.0171) [3]−	0.9413 (0.0511) [2]−	0.1733 (0.0260) [4]−	0.1106 (0.0056) [5]−	0.9949 (0.0103) [1]
DLTZ6	0.9900 (0.0025) [5]−	0.9973 (0.0008) [3]≈	0.9997 (0.0005) [1]≈	0.979 (0.0057) [6]−	0.9944 (0.0146) [4]−	0.9984 (0.0008) [2]
DLTZ7	0.8767 (0.0306) [5]−	0.8489 (0.0643) [6]−	0.9895 (0.0074) [2]≈	0.9898 (0.0008) [1]≈	0.9389 (0.0952) [4]−	0.9894 (0.0330) [3]
Mean rank	4.14	3.71	1.57	4.00	4.85	2.71
+ / − / ≈	2/4/1	2/4/1	4/1/2	0/5/2	0/7/0	—

Table 3. Statistics (mean (std. dev.)) of Contribution metric values of the final populations obtained by MOLAPO, GS, MOPSO, NSGA-II, MNMA, and MOMmCAA over 30 independent runs on the DLTZ benchmark.

Fn	MOLAPO	GS	MOPSO	NGSGA-II	MNMA	MOMmCAA
DLTZ1	0.815 (2.5×10^{-16}) [2]−	0.797 (1.8×10^{-16}) [5]−	0.814 (2.6×10^{-16}) [3]−	0.8095 (0.0124) [4]−	0.5406 (0.0061) [6]−	0.999 (1.5×10^{-16}) [1]
DLTZ2	0.6088 (0.0461) [5]−	0.7600 (0.0644) [3]−	0.9908 (0.0404) [1]+	0.6421 (0.1655) [4]−	0.1969 (0.0361) [6]−	0.9248 (0.0116) [2]
DLTZ3	0.815 (6.8×10^{-17}) [2]−	0.797 (2.5×10^{-16}) [5]−	0.814 (6.8×10^{-17}) [3]−	0.8120 (0.0087) [4]−	0.4612 (0.0301) [6]−	0.997 (2.3×10^{-17}) [1]
DLTZ4	0.810 (6.9×10^{-17}) [3]−	0.797 (1.5×10^{-16}) [5]−	0.814 (6.9×10^{-17}) [2]−	0.804 (7.9×10^{-17}) [4]−	0.6655 (0.0641) [6]−	0.966 (1.9×10^{-17}) [1]
DLTZ5	0.0011 (0.0056) [6]−	0.1416 (0.0179) [4]−	0.8995 (0.1141) [2]−	0.1619 (0.2573) [3]−	0.0071 (0.0025) [5]−	0.9135 (0.1269) [1]
DLTZ6	0.0793 (0.0001) [6]−	0.1285 (0.0010) [5]−	0.3526 (0.0168) [3]−	0.9791 (0.0003) [1]+	0.2968 (0.0908) [4]−	0.5203 (0.0135) [2]
DLTZ7	0.1476 (0.0034) [5]−	0.0976 (0.0205) [6]−	0.7690 (0.0170) [3]−	0.7783 (0.0327) [2]−	0.7655 (0.0684) [4]−	0.9489 (0.0001) [1]
Mean rank	4.14	4.71	2.42	3.14	5.28	1.28
+ / − / ≈	0/7/0	0/7/0	1/6/0	1/6/0	0/7/0	—

Table 4. Statistics (mean (std. dev.)) of Epsilon Indicator metric values of the final populations obtained by MOLAPO, GS, MOPSO, NSGA-II, MNMA, and MOMmCAA over 30 independent runs on the DLTZ benchmark.

Fn	MOLAPO	GS	MOPSO	NGSGA-II	MNMA	MOMmCAA
DLTZ1	0.9925 (0.0064) [3]≈	0.9963 (0.0032) [2]+	0.9994 (0.0015) [1]+	0.8791 (0.0395) [6]−	0.9655 (0.0140) [5]−	0.9924 (0.0053) [4]
DLTZ2	0.9936 (0.0060) [3]−	0.9731 (0.0089) [6]−	0.9985 (0.0015) [1]≈	0.9931 (0.0070) [4]−	0.9819 (0.1230) [5]−	0.9984 (0.0059) [2]
DLTZ3	0.9555 (0.0241) [4]−	0.9702 (0.0249) [2]+	0.9995 (0.0020) [1]+	0.6953 (0.0780) [6]−	0.7779 (0.0963) [5]−	0.9689 (0.0217) [3]
DLTZ4	0.8053 (0.1128) [4]−	0.9047 (0.0235) [2]+	0.9978 (0.0069) [1]+	0.7507 (0.1543) [5]−	0.7380 (0.1501) [6]−	0.8434 (0.1174) [3]
DLTZ5	0.8790 (0.0086) [5]−	0.737 (2.9×10^{-10}) [6]−	0.911 (3.3×10^{-10}) [4]−	0.979 (8.6×10^{-10}) [2]−	0.921 (3.4×10^{-10}) [3]−	0.999 (4.9×10^{-11}) [1]
DLTZ6	0.8967 (0.0500) [5]−	0.9276 (0.0021) [4]−	0.9991 (0.0002) [1]+	0.4459 (0.1339) [6]−	0.9687 (0.0611) [2]≈	0.9684 (0.0228) [3]
DLTZ7	0.8996 (0.0315) [6]−	0.9427 (0.0038) [5]−	0.9957 (0.0072) [1]+	0.9429 (0.0944) [4]−	0.9689 (0.0142) [3]≈	0.9694 (0.0146) [2]
Mean rank	4.28	3.85	1.43	4.71	4.14	2.57
+ / − / ≈	0/6/1	3/4/0	5/1/1	0/7/0	0/5/2	—

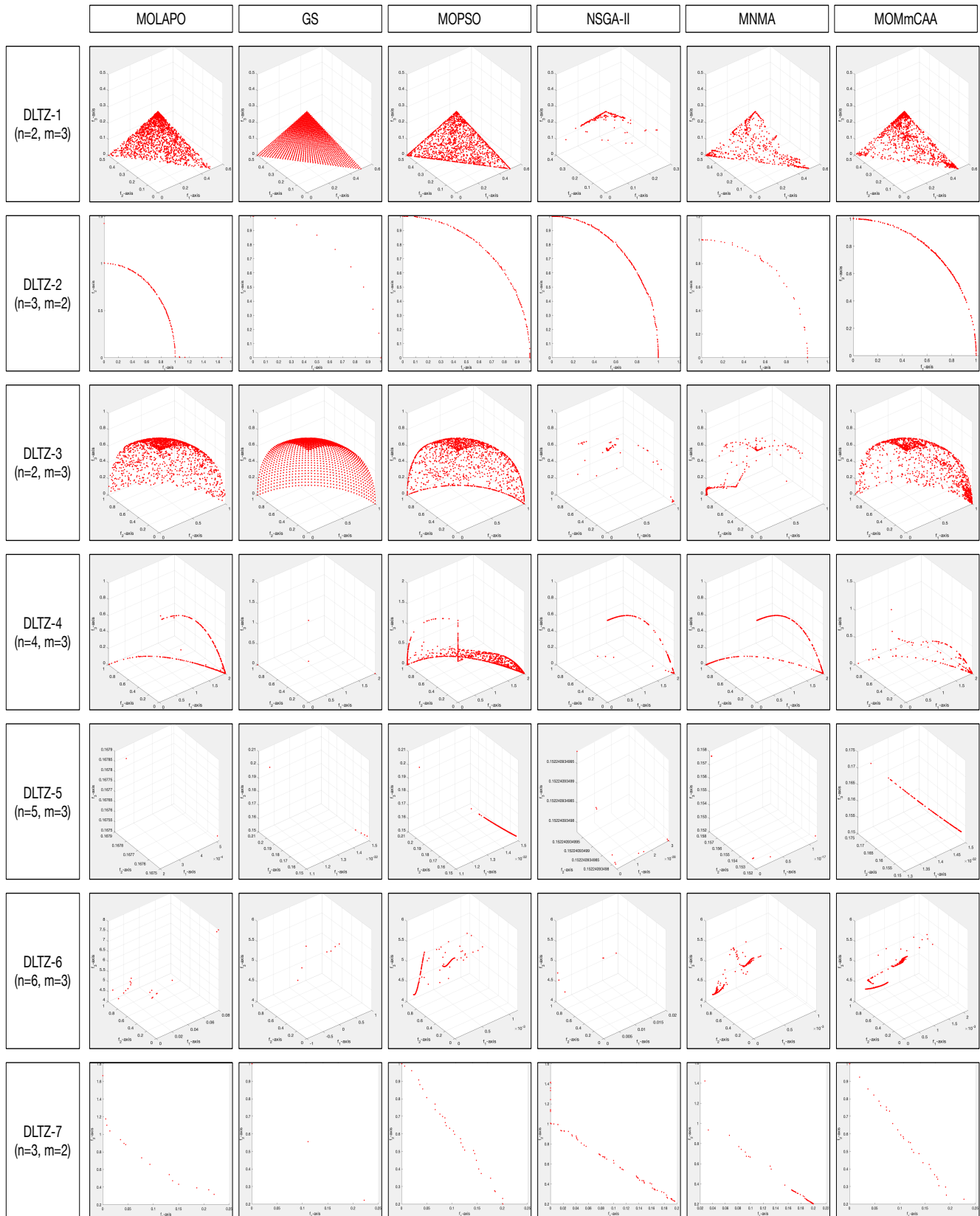


Figure 4. Representative PFs obtained by the seven MOEAs on the DLTZ benchmark.

3.3. Quadratic Instances

The Quadratics test set is a randomly generated test set described in [50]. The objective functions are all of the form $\frac{1}{2}x^T Ax + bx + c$, where the components A , b , and c are random numbers in the range $[-1, 1]$. A is not a symmetric matrix, and the test set is non-convex. Tables 5–7 expose the results of the metric values obtained by the algorithms over the ten quadratic problems.

Table 5 shows that the MOMmCAA obtains significantly better *HV* values than MOLAPO, GS, MOPSO, NSGA-II, and MNMA for six, seven, two, eight, and seven out of the eight instances, respectively. Regarding the overall mean rankings, the MOMmCAA obtains the second optimal mean rank value after MOPSO, followed by the other algorithms. The MOMmCAA demonstrates performance that is significantly equivalent to MOPSO regarding the other three instances. Table 6 shows that the MOMmCAA achieves ten, ten, five, eight, and seven better *C* metric values than MOLAPO, GS, MOPSO, NSGA-II, and MNMA, respectively. This indicates the quality of the solutions obtained by the MOMmCAA. Table 7 depicts the overall performance of the six algorithms in terms of their *EI* metric values. The MOMmCAA yields significantly better *EI* values than MOLAPO, GS, MOPSO, NSGA-II, and MNMA for nine, seven, six, eight, and nine out of the ten instances, respectively. Figure 5 shows representative Pareto fronts (PFs) obtained by the six comparison MOEAs. In summary, the MOMmCAA shows competitive performance on the Quadratic benchmark.

3.4. CEC2020 Instances

Tables 8–10 present the results of the metric values obtained by the algorithms on ten benchmark CEC2020 problems.

Table 8 shows that the MOMmCAA obtains significantly better *HV* values than MOLAPO, GS, MOPSO, NSGA-II, and MNMA for eight, nine, four, eight, and seven out of the ten instances, respectively. In the case of MOPSO, there are six results with no significant difference. Concerning the overall mean rankings, the MOMmCAA obtains the optimal mean rank value. The MOMmCAA demonstrates poor performance on the MMF-2 and MMF-7 test instances. In summary, the MOMmCAA is superior to all the other MOEAs in terms of this metric. Table 9 shows that the MOMmCAA achieves ten, ten, seven, eight, and six better *C* metric values than MOLAPO, GS, MOPSO, NSGA-II, and MNMA, respectively. In the case of MNMA, there are four results with the worst significant difference. Table 10 summarizes the overall performance of the six algorithms in terms of their *EI* metric values. The MOMmCAA yields significantly better *EI* values than MOLAPO, GS, MOPSO, NSGA-II, and MNMA for ten, ten, six, ten, and eight out of the ten instances, respectively. Figure 6 depicts the representative Pareto fronts (PFs) obtained by the seven comparison MOEAs. In summary, the MOMmCAA shows competitive behavior on the CEC2020 benchmark.

Table 5. Statistics (mean (std. dev.)) of Hypervolume metric values of the final populations obtained by MOLAPO, GS, MOPSO, NSGA-II, MNMA, and MOMmCAA over 30 independent runs on the Quadratic test suite.

Fn	MOLAPO	GS	MOPSO	NSGA-II	MNMA	MOMmCAA
Quad-1	0.7832 (0.0465) [5]−	0.9192 (0.2361) [3]−	0.9999 (0.0105) [1]+	0.9008 (0.0359) [4]−	0.7501 (0.1797) [6]−	0.9730 (0.0100) [2]
Quad-2	0.7556 (0.0364) [6]−	0.7834 (0.2361) [5]−	0.9942 (0.0986) [1]+	0.8271 (0.0389) [4]−	0.9267 (0.1588) [3]−	0.9668 (0.0118) [2]
Quad-3	0.9828 (0.0038) [5]−	0.9744 (0.0040) [6]−	0.9901 (0.0034) [4]−	0.9983 (0.0022) [1]+	0.9954 (0.0041) [2]≈	0.9953 (0.0050) [3]
Quad-4	0.9988 (0.0017) [1]≈	0.9957 (0.0046) [3]≈	0.9911 (0.0032) [4]−	0.9836 (0.0037) [5]−	0.9751 (0.0048) [6]−	0.9969 (0.0029) [2]
Quad-5	0.9409 (0.0108) [4]≈	0.9232 (0.0164) [5]−	0.9999 (0.0044) [1]+	0.9932 (0.0037) [2]+	0.8947 (0.0141) [6]−	0.9687 (0.0059) [3]
Quad-6	0.9834 (0.0050) [4]≈	0.9674 (0.0082) [6]−	0.9876 (0.0072) [3]≈	0.9759 (0.0114) [5]−	0.9993 (0.0015) [1]≈	0.9965 (0.0038) [2]
Quad-7	0.4328 (0.2074) [6]−	0.4650 (0.0808) [5]−	0.9880 (0.0653) [1]+	0.7888 (0.0595) [3]−	0.5682 (0.1853) [4]−	0.9374 (0.0143) [2]
Quad-8	0.9929 (0.0042) [4]≈	0.9983 (0.0028) [1]≈	0.9930 (0.0041) [3]≈	0.9649 (0.0151) [5]−	0.9622 (0.0204) [6]−	0.9947 (0.0058) [2]
Quad-9	0.9692 (0.0092) [4]−	0.9788 (0.0061) [3]−	0.9977 (0.0010) [1]≈	0.9010 (0.0203) [6]−	0.9077 (0.0257) [5]−	0.9973 (0.0058) [2]
Quad-10	0.8677 (0.0117) [4]−	0.9744 (0.0046) [2]+	0.9814 (0.0055) [1]+	0.7300 (0.0677) [6]−	0.8431 (0.0594) [5]−	0.9210 (0.0101) [3]
Mean rank	4.30	3.90	2.00	4.10	4.40	2.30
+ / − / ≈	0/6/4	1/7/2	5/2/3	2/8/0	0/8/2	—

Table 6. Statistics (mean (std. dev.)) of Contribution metric values of the final populations obtained by MOLAPO, GS, MOPSO, NSGA-II, MNMA, and MOMmCAA over 30 independent runs on the Quadratic test suite.

Fn	MOLAPO	GS	MOPSO	NGSGA-II	MNMA	MOMmCAA
Quad-1	0.7548 (0.0227) [4]−	0.8559 (0.3276) [3]−	0.9938 (0.0237) [1]+	0.6563 (0.0716) [5]−	0.6304 (0.0708) [6]−	0.9158 (0.0698) [2]
Quad-2	0.6150 (0.0140) [6]−	0.7628 (0.1674) [4]−	0.9189 (0.1903) [1]≈	0.7564 (0.2903) [5]−	0.8209 (0.0154) [3]−	0.9181 (0.1407) [2]
Quad-3	0.5834 (0.0181) [5]−	0.5745 (0.0118) [6]−	0.6594 (0.0327) [4]−	0.9827 (0.0828) [1]+	0.8591 (0.1329) [2]+	0.7498 (0.0357) [3]
Quad-4	0.9591 (0.1329) [2]−	0.9498 (0.0357) [3]−	0.8594 (0.0227) [4]−	0.6745 (0.0818) [6]−	0.6834 (0.0181) [5]−	0.9827 (0.0823) [1]
Quad-5	0.8493 (0.0419) [4]−	0.6621 (0.0211) [5]−	0.9848 (0.0452) [1]+	0.9398 (0.1482) [2]+	0.6382 (0.0162) [6]−	0.8862 (0.0741) [3]
Quad-6	0.8290 (0.0486) [3]−	0.6627 (0.0202) [6]−	0.7066 (0.0408) [4]−	0.6643 (0.0172) [5]−	0.9965 (0.0130) [1]+	0.9176 (0.0761) [2]
Quad-7	0.5266 (0.0516) [5]−	0.4667 (0.0821) [6]−	0.9711 (0.0453) [1]+	0.8471 (0.2180) [3]−	0.7424 (0.0153) [4]−	0.9264 (0.0303) [2]
Quad-8	0.7772 (0.0072) [4]−	0.8870 (0.0048) [2]−	0.8073 (0.0092) [3]−	0.6046 (0.0122) [5]−	0.5718 (0.0162) [6]−	0.9951 (0.0031) [1]
Quad-9	0.5452 (0.0215) [5]−	0.4369 (0.0118) [6]−	0.8706 (0.0423) [2]−	0.6800 (0.0718) [4]−	0.8547 (0.0887) [3]−	0.9990 (0.0051) [1]
Quad-10	0.6170 (0.0055) [6]−	0.6607 (0.0112) [5]−	0.9051 (0.0803) [2]+	0.7536 (0.1651) [4]−	0.9722 (0.0694) [1]+	0.0585 (0.8018) [3]
Mean rank	4.40	4.60	2.30	4.00	3.70	2.00
+ / − / ≈	0/10/0	0/10/0	4/5/1	2/8/0	3/7/0	—

Table 7. Statistics (mean (std. dev.)) of Epsilon Indicator metric values of the final populations obtained by MOLAPO, GS, MOPSO, NSGA-II, MNMA, and MOMmCAA over 30 independent runs on the Quadratic test suite.

Fn	MOLAPO	GS	MOPSO	NSGA-II	MNMA	MOMmCAA
Quad-1	0.6240 (0.0944) [5]−	0.7748 (0.1275) [3]−	0.9951 (0.0044) [2]≈	0.6962 (0.3369) [4]−	0.5028 (0.2753) [6]−	0.9981 (0.0036) [1]
Quad-2	0.5449 (0.0991) [6]−	0.8027 (0.2635) [3]−	0.9879 (0.0244) [1]+	0.5542 (0.2826) [5]−	0.5996 (0.1161) [4]−	0.9779 (0.0194) [2]
Quad-3	0.8620 (0.1774) [5]−	0.7980 (0.1186) [6]−	0.8751 (0.1856) [4]−	0.9776 (0.0347) [1]+	0.9108 (0.1406) [3]−	0.9432 (0.0674) [2]
Quad-4	0.9783 (0.0343) [1]+	0.9181 (0.1343) [3]−	0.7756 (0.1862) [4]−	0.3984 (0.1191) [6]−	0.4685 (0.1619) [5]−	0.9439 (0.0679) [2]
Quad-5	0.8844 (0.1060) [4]−	0.4184 (0.0458) [6]−	0.9909 (0.0175) [1]+	0.9082 (0.0663) [3]−	0.4975 (0.0550) [5]−	0.9383 (0.0701) [2]
Quad-6	0.6550 (0.1195) [6]−	0.7024 (0.1305) [5]−	0.7376 (0.1019) [4]−	0.8806 (0.0519) [3]≈	0.9658 (0.0430) [1]+	0.8999 (0.0771) [2]
Quad-7	0.5581 (0.0323) [5]−	0.5366 (0.1128) [6]−	0.9571 (0.1227) [2]−	0.7736 (0.1198) [3]−	0.6356 (0.1532) [4]−	0.9747 (0.0266) [1]
Quad-8	0.9234 (0.0884) [3]−	0.9878 (0.0524) [1]+	0.9188 (0.0135) [4]−	0.8119 (0.0454) [6]−	0.8986 (0.0560) [5]−	0.9488 (0.0476) [2]
Quad-9	0.6500 (0.1554) [6]−	0.9671 (0.1128) [1]+	0.9536 (0.0220) [2]≈	0.7165 (0.1319) [5]−	0.7200 (0.2024) [4]−	0.9484 (0.1139) [3]
Quad-10	0.6713 (0.0765) [4]−	0.9971 (0.0120) [1]+	0.9174 (0.0983) [2]+	0.5120 (0.0297) [6]−	0.5753 (0.0895) [5]−	0.7920 (0.0212) [3]
Mean rank	4.50	3.50	2.60	4.20	4.20	2.00
+ / − / ≈	1/9/0	3/7/0	2/6/2	1/8/1	1/9/0	—

Table 8. Statistics (mean (std. dev.)) of Hypervolume metric values of the final populations obtained by MOLAPO, GS, MOPSO, NSGA-II, MNMA, and MOMmCAA over 30 independent runs on the CEC2020 test suite.

Fn	MOLAPO	GS	MOPSO	NSGA-II	MNMA	MOMmCAA
MMF-1	0.9955 (0.0038) [2]≈	0.9792 (0.0043) [5]−	0.9952 (0.0036) [3]≈	0.9850 (0.0100) [4]−	0.9533 (0.0468) [6]−	0.9997 (0.0006) [1]
MMF-2	0.9451 (0.2137) [3]≈	0.9451 (0.2139) [5]≈	0.9453 (0.2134) [2]≈	0.9446 (0.2154) [6]−	0.9732 (0.2031) [1]+	0.9451 (0.2138) [4]
MMF-4	0.9276 (0.3679) [3]−	0.9171 (0.3664) [4]−	0.8989 (0.3692) [5]−	0.8378 (0.3683) [6]−	0.9785 (0.2451) [2]−	0.9983 (0.3689) [1]
MMF-5	0.9906 (0.0088) [4]−	0.9851 (0.0072) [6]−	0.9918 (0.0077) [3]−	0.9865 (0.0112) [5]−	0.9962 (0.0048) [2]≈	0.9976 (0.0042) [1]
MMF-7	0.9736 (0.0431) [5]−	0.9662 (0.0398) [6]−	0.9755 (0.0419) [2]≈	0.9754 (0.0425) [4]≈	0.9968 (0.0481) [1]+	0.9755 (0.0417) [3]
MMF-8	0.9116 (0.0631) [4]−	0.9102 (0.0591) [5]−	0.9555 (0.0519) [2]−	0.9324 (0.0521) [3]−	0.3913 (0.0531) [6]−	0.9713 (0.0466) [1]
MMF-10	0.8607 (0.0026) [6]−	0.9331 (0.0489) [4]−	0.9602 (0.0238) [3]−	0.9793 (0.0029) [1]≈	0.9055 (0.0967) [5]−	0.9721 (0.0223) [2]
MMF-11	0.8512 (0.0338) [6]−	0.8627 (0.0011) [5]−	0.9723 (0.0211) [1]≈	0.9278 (0.0887) [3]−	0.8891 (0.0261) [4]−	0.9715 (0.0018) [2]
MMF-12	0.9618 (0.0038) [5]−	0.9319 (0.0842) [6]−	0.9973 (0.0034) [2]≈	0.9639 (0.0138) [4]−	0.9773 (0.0094) [3]−	0.9987 (0.0029) [1]
MMF-13	0.9068 (0.0054) [6]−	0.9192 (0.0026) [5]−	0.9991 (0.0017) [1]≈	0.9788 (0.0049) [3]−	0.9689 (0.0247) [4]−	0.9982 (0.0020) [2]
Mean rank	4.4	5.1	2.4	3.9	3.2	1.8
+ / − / ≈	0/8/2	0/9/1	0/4/6	0/8/2	2/7/1	—

Table 9. Statistics (mean (std. dev.)) of Contribution metric values of the final populations obtained by MOLAPO, GS, MOPSO, NSGA-II, MNMA, and MOMmCAA over 30 independent runs on the CEC2020 test suite.

Fn	MOLAPO	GS	MOPSO	NGSGA-II	MNMA	MOMmCAA
MMF-1	0.2128 (0.0289) [6]−	0.4868 (0.0616) [3]−	0.2676 (0.0319) [5]−	0.3151 (0.0662) [4]−	0.9763 (0.0461) [1]+	0.8734 (0.1375) [2]
MMF-2	0.2715 (0.0014) [6]−	0.3240 (0.0122) [4]−	0.3588 (0.0870) [3]−	0.8086 (0.0059) [5]−	0.9993 (0.1599) [1]+	0.9369 (0.1297) [2]
MMF-4	0.1328 (0.0583) [6]−	0.5403 (0.2403) [3]−	0.2297 (0.1104) [5]−	0.3263 (0.1457) [4]−	0.8489 (0.2181) [2]−	0.8717 (0.2841) [1]
MMF-5	0.1913 (0.0175) [6]−	0.3997 (0.0524) [3]−	0.2309 (0.0374) [5]−	0.3514 (0.0576) [4]−	0.9977 (0.0074) [1]+	0.9444 (0.0169) [2]
MMF-7	0.1712 (0.2828) [6]−	0.7725 (0.0934) [3]−	0.3623 (0.0616) [5]−	0.6166 (0.2130) [4]−	0.9838 (0.1464) [1]+	0.9029 (0.1444) [2]
MMF-8	0.3752 (0.0038) [5]−	0.8731 (0.0093) [3]−	0.8623 (0.0061) [4]−	0.9153 (0.0028) [1]+	0.1718 (0.0014) [6]−	0.9011 (0.0014) [2]
MMF-10	0.8731 (0.0420) [6]−	0.9011 (0.0285) [4]−	0.9302 (0.0131) [3]−	0.9561 (0.0023) [2]≈	0.8925 (0.0367) [5]−	0.9573 (0.0019) [1]
MMF-11	0.6796 (0.0416) [6]−	0.7734 (0.0183) [5]−	0.9407 (0.0223) [1]+	0.9167 (0.0071) [3]−	0.7992 (0.0331) [4]−	0.9366 (0.0112) [2]
MMF-12	0.5151 (0.0734) [5]−	0.4905 (0.0204) [6]−	0.9854 (0.0601) [2]≈	0.6131 (0.1622) [3]−	0.5951 (0.2830) [4]−	0.9866 (0.0511) [1]
MMF-13	0.5081 (0.0882) [6]−	0.6111 (0.0613) [5]−	0.9618 (0.0883) [1]+	0.8055 (0.0775) [4]−	0.8305 (0.0798) [3]−	0.9189 (0.0613) [2]
Mean rank	5.8	3.9	3.4	3.4	2.8	1.7
+ / − / ≈	0/10/0	0/10/0	2/7/1	1/8/1	4/6/0	—

Table 10. Statistics (mean (std. dev.)) of Epsilon Indicator metric values of the final populations obtained by MOLAPO, GS, MOPSO, NSGA-II, MNMA, and MOMmCAA over 30 independent runs on the CEC2020 test suite.

Fn	MOLAPO	GS	MOPSO	NGSGA-II	MNMA	MOMmCAA
MMF-1	0.9787 (0.0493) [3]−	0.9528 (0.0410) [4]−	0.9802 (0.0472) [2]≈	0.9349 (0.0475) [5]−	0.8351 (0.1070) [6]−	0.9832 (0.0498) [1]
MMF-2	0.9586 (0.0293) [3]−	0.9328 (0.0209) [4]−	0.9602 (0.0272) [2]≈	0.9149 (0.0275) [5]−	0.7652 (0.1236) [6]−	0.9632 (0.0298) [1]
MMF-4	0.6585 (0.0393) [6]−	0.8386 (0.0229) [3]−	0.6602 (0.0282) [5]≈	0.7149 (0.0295) [4]−	0.8402 (0.1036) [2]−	0.8632 (0.0197) [1]
MMF-5	0.8870 (0.1226) [4]−	0.8793 (0.1163) [5]−	0.8922 (0.1229) [3]−	0.8787 (0.1204) [6]−	0.9858 (0.0285) [1]≈	0.9851 (0.0124) [2]
MMF-7	0.8289 (0.1358) [6]−	0.8314 (0.1339) [4]−	0.8345 (0.1374) [3]−	0.8310 (0.1347) [5]−	0.9988 (0.0466) [1]+	0.9333 (0.0137) [2]
MMF-8	0.7739 (0.0043) [5]−	0.7900 (0.0039) [4]−	0.8999 (0.0002) [1]≈	0.8696 (0.0038) [3]−	0.3130 (0.0389) [6]−	0.8994 (0.0010) [2]
MMF-10	0.8611 (0.0041) [6]−	0.8918 (0.0081) [5]−	0.9192 (0.0113) [4]≈	0.9351 (0.0066) [2]−	0.9302 (0.0023) [3]−	0.9416 (0.0031) [1]
MMF-11	0.6951 (0.0164) [6]−	0.7959 (0.0217) [5]−	0.9023 (0.0086) [1]+	0.8365 (0.0043) [3]−	0.8052 (0.0065) [4]−	0.8974 (0.0019) [2]
MMF-12	0.7487 (0.2003) [6]−	0.9148 (0.0109) [5]−	0.9892 (0.0113) [2]≈	0.9280 (0.0905) [4]−	0.9529 (0.0356) [3]−	0.9956 (0.0125) [1]
MMF-13	0.9369 (0.0115) [5]−	0.8624 (0.1193) [6]−	0.9965 (0.0101) [1]+	0.9458 (0.0317) [4]−	0.9575 (0.0244) [3]−	0.9754 (0.0301) [2]
Mean rank	4.8	4.5	2.4	4.1	3.5	1.5
+ / − / ≈	0/10/0	0/10/0	2/6/2	0/10/0	1/8/1	—

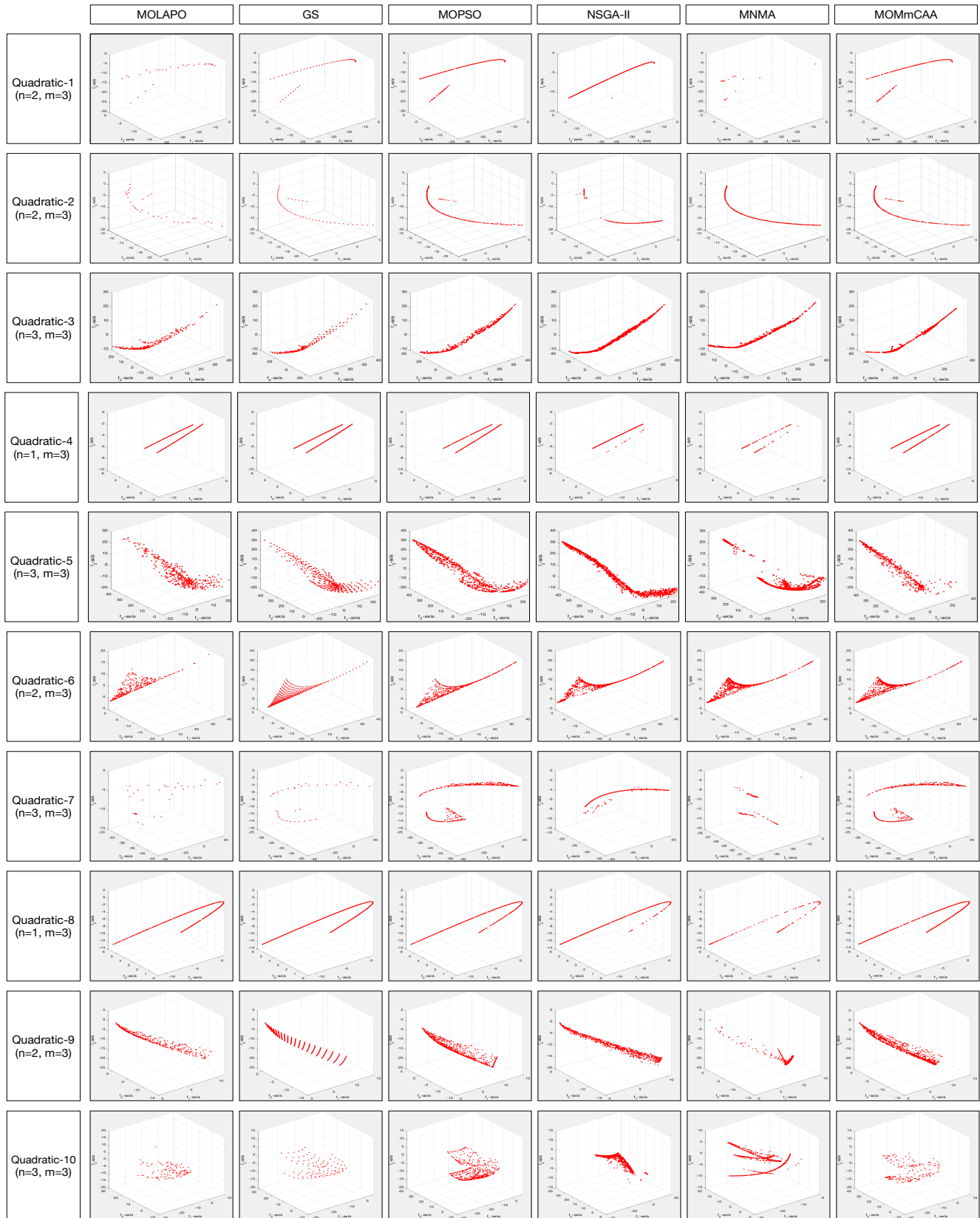


Figure 5. Representative PFs obtained by the six MOEAs on the Quadratic benchmark set.

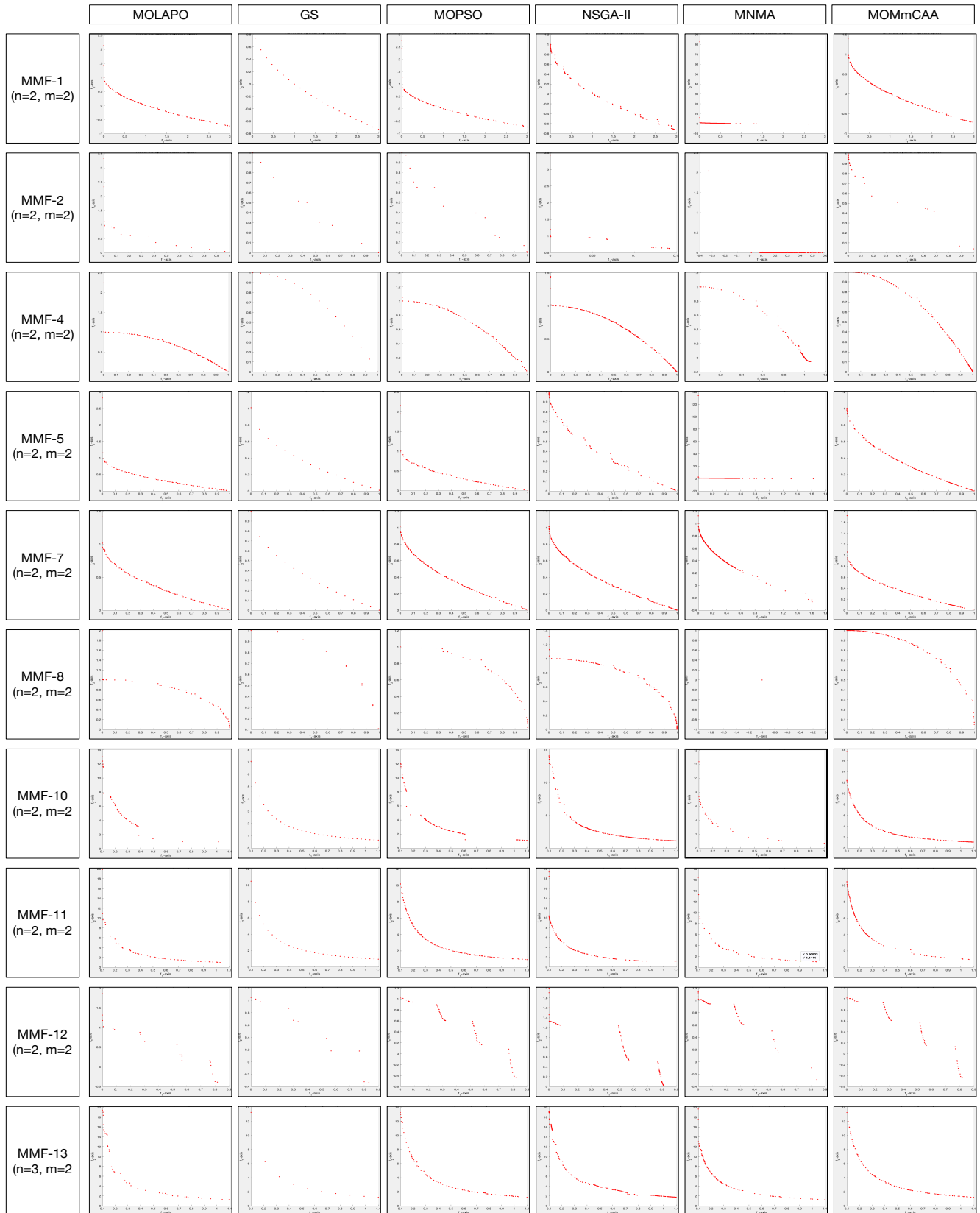


Figure 6. Representative PFs obtained by the six MOEAs on the CEC2020 benchmark set.

4. Engineering Design Problems

In this section, the capability of the MOMmCAA is evaluated in solving two real-world engineering design problems: design of a four-bar truss and design of a disk brake.

4.1. Four-Bar Truss Design Problem

In the four-bar truss design [58], the structural volume h_1 and displacement h_2 have to be minimized. This problem consists of four design variables z_1 to z_4 corresponding to the cross-sectional area of parts 1 to 4, as illustrated in Figure 7. The equations are provided below.

Minimize:

$$h_1(z) = 200(2z_1 + \sqrt{2z_2} + \sqrt{z_3} + z_4)$$

$$h_2(z) = 0.01 \left(\frac{2}{z_1} + \frac{2\sqrt{2}}{z_2} - \frac{2\sqrt{2}}{z_3} + \frac{2}{z_4} \right) \quad (6)$$

where:

$$1 \leq z_1 \leq 3, 1.4142 \leq z_2 \leq 3$$

$$1.4142 \leq z_3 \leq 3, 1 \leq z_4 \leq 3$$

This problem involves minimizing two components, h_1 and h_2 . One approach is to combine both functions using $h(z) = \lambda_1 h_1(z) + \lambda_2 h_2(z)$ or $h(z) = h_1^{q_1}(z) + h_2^{q_2}(z)$, where λ_i and q_i are the weighting coefficients [59]. However, determining these coefficients relies on experience and a trial-and-error process to achieve the desired results. Therefore, the multi-objective approach considers both functions separately in order to approximate the PF and obtain multiple non-dominated solutions.

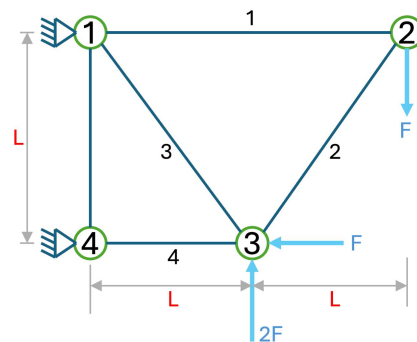


Figure 7. Description of the four-bar truss design problem.

4.2. Disk Brake Design Problem

The multi-objective disc brake design problem proposed in [60] has five constraints and two objectives to be minimized, namely, the stopping time h_1 and brake mass h_2 . This problem has four design variables: the inner radius of the disc z_1 , the outer radius z_2 , the engaging force z_3 , and the number of friction surfaces z_4 . Figure 8 depicts the system and Equation (7) describes the problem.

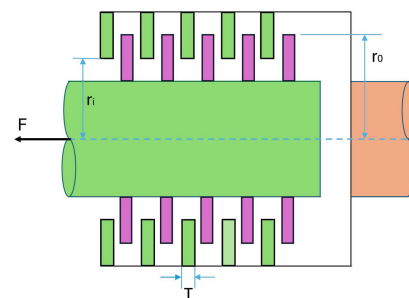


Figure 8. Description of the disk brake design problem.

Minimize:

$$h_1(z) = (4.9 * 10^{-5})(z_2^2 - z_1^2)(z_4 - 1)$$

$$h_2(z) = (9.82 * 10^6) \frac{z_2^2 - z_1^2}{(z_2^3 - z_1^3)(z_4 z_3)}$$

Subject to:

$$g_1(z) = z_2 - z_1 - 20$$

$$g_2(z) = 30 - (2.5(z_4 - 1))$$

$$g_3(z) = 0.4 - \frac{z_3}{3.14 * (z_2^2 - z_1^2)}$$

$$g_4(z) = 1 - (2.22 * 10^{-3}) \frac{z_3(z_2^3 - z_1^3)}{(z_2^2 - z_1^2)^2}$$

$$g_5(z) = (2.66 * 10^{-2}) \frac{z_3 z_4 (z_2^3 - z_1^3)}{(z_2^2 - z_1^2)} - 900$$

(7)

where:

$$55 \leq z_1 \leq 80, 75 \leq z_2 \leq 110$$

$$1000 \leq z_3 \leq 3000, 2 \leq z_4 \leq 20$$

4.3. Design Problem Results

Table 11 presents the statistical results of the MOMmCAA and the other algorithms in dealing with the engineering design problems using the performance metrics of *HV*, *C* and *EI*. It can be seen that the MOMmCAA is one of the two best algorithms for these metrics in both cases, demonstrating the competitiveness of the proposed algorithm. The MOMmCAA is able to calculate better results than MOLAPO, GS, NSGA-II, and MNMA, and its results are very close to those obtained with MOPSO. Figure 9 presents the PFs obtained by the different algorithms for the two engineering design problems.

Table 11. Statistics (mean (std. dev.)) of all metric values of the final populations obtained by MOLAPO, GS, MOPSO, NSGA-II, MNMA, and MOMmCAA over 30 independent runs on the four-bar truss and disk brake design problems.

Four-Bar Truss	MOLAPO	GS	MOPSO	NSGA-II	MNMA	MOMmCAA
HV	0.8459 (0.0123) [6]	0.8719 (0.0234) [5]	0.9598 (0.0041) [2]	0.9370 (0.0098) [3]	0.9182 (0.0377) [4]	0.9678 (0.0026) [1]
C	0.2666 (0.0146) [6]	0.3927 (0.0725) [5]	0.9790 (0.0704) [1]	0.7765 (0.3033) [3]	0.7477 (0.0203) [4]	0.9681 (0.0641) [2]
EI	0.5127 (0.0605) [6]	0.5824 (0.0274) [5]	0.9204 (0.0389) [1]	0.8913 (0.2059) [3]	0.8603 (0.2389) [4]	0.9184 (0.0270) [2]
Disk Brake	MOLAPO	GS	MOPSO	NSGA-II	MNMA	MOMmCAA
HV	0.8780 (0.0180) [5]	0.8737 (0.0051) [6]	0.9947 (0.0024) [1]	0.9857 (0.0023) [4]	0.9923 (0.0027) [3]	0.9931 (0.0012) [2]
C	0.6498 (0.0882) [5]	0.5346 (0.1352) [6]	0.9128 (0.2202) [2]	0.8296 (0.2406) [3]	0.7847 (0.3748) [4]	0.9128 (0.2202) [1]
EI	0.7637 (0.0608) [5]	0.7329 (0.0252) [6]	0.9614 (0.0092) [2]	0.9216 (0.0325) [3]	0.8936 (0.0888) [4]	0.9971 (0.0080) [1]

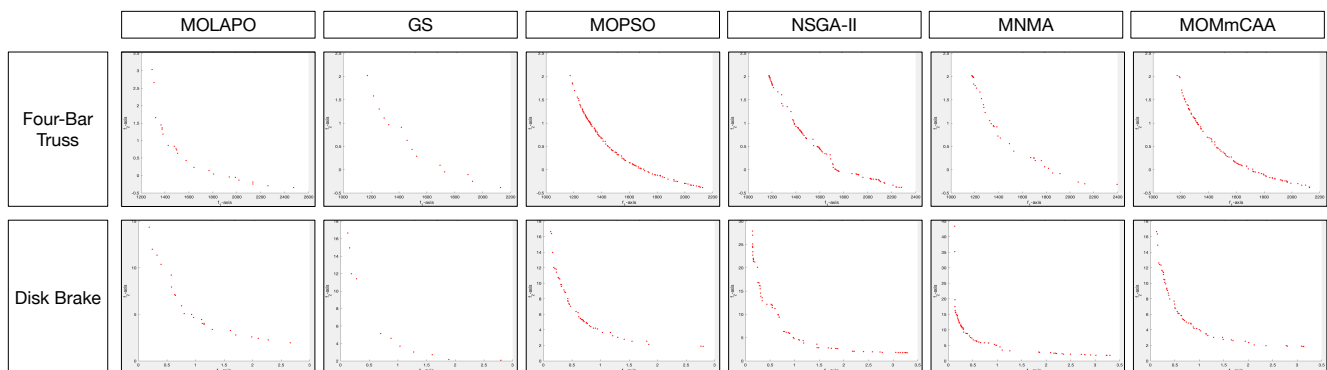


Figure 9. Representative PFs obtained by the six MOEAs on the two engineering design problems.

5. Conclusions and Further Work

This paper presents a new multi-objective optimization algorithm called the MOMmCAA inspired by the neighborhood and local interaction rules of majority and minority cellular automata. The randomness, concurrency, and information exchange generated between the smart-cells by applying different rules produce an appropriate balance between exploration and exploitation actions.

Comparative computational testing was carried out on 27 test functions with various characteristics, including convex, concave, mixed, disconnected, and degenerated PFs. These test functions were used to challenge the MOMmCAA, and its performance was compared against five other algorithms recognized for their efficiency. The experiments showed satisfactory performance on the part of the MOMmCAA.

In addition, two multi-objective engineering problems from the recent literature were used to test the MOMmCAA against the results obtained by the other algorithms. The MOMmCAA again demonstrated its high quality in finding solutions to these problems, proving its competitiveness against other recent metaheuristics.

Compared to classical techniques, the MOMmCAA provides improved flexibility. It can explore large search spaces and adapt to problems with multiple objectives and complex constraints. These features make the MOMmCAA especially useful for solving multi-objective optimization problems, where traditional methods may be inefficient due to assumptions about the problem's nature, the need for derivatives, or the complexity of the objective functions.

As further work, the MOMmCAA has to be proven effective in solving real-world problems such as power grid design, vehicle routing optimization, industrial systems control, and feature selection in bioinformatics. Its ability to balance multiple conflicting criteria makes it suitable for multi-objective situations.

However, the MOMmCAA has limitations in scalability for high-dimensional problems, where managing the repository of non-dominated solutions and correctly selecting the algorithm parameters are critical aspects affecting its performance. Its computational cost can also be high when dealing with complex problems, especially when requiring many iterations or accurate PF estimation.

These limitations provide opportunities for future algorithm refinement, including testing improvements with fewer parameters, dynamic parameter control, or other solution control mechanisms such as niche strategy, clustering, rank dominance, or FP maintenance methods. The richness of cellular automata behaviors also presents new opportunities for proposing new multi-objective optimization algorithms, such as the utilization of periodic, chaotic, universal, complex, or surjective and reversible cellular automata.

Author Contributions: Conceptualization, J.C.S.-T.-M. and J.M.-M.; methodology, J.C.S.-T.-M. and U.H.-H.; validation, J.C.S.-T.-M. and J.M.-M.; formal analysis, J.C.S.-T.-M., L.L.-M. and N.H.-R.; investigation, J.C.S.-T.-M., U.H.-H. and J.M.-M.; resources, N.H.-R. and L.L.-M.; writing—original draft preparation, J.C.S.-T.-M., U.H.-H. and J.M.-M.; writing—review and editing, N.H.-R. and L.L.-M.; visualization, J.C.S.-T.-M. and N.H.-R.; supervision, J.M.-M. and L.L.-M.; funding acquisition, J.C.S.-T.-M. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by the Autonomous University of Hidalgo (UAEH) and the National Council for Humanities, Science and Technology (CONAHCYT) with project number F003-320109.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The MOMmCAA source code is available on Github <https://github.com/juanseck/MOMmCAA> (accessed on 23 August 2024).

Conflicts of Interest: The authors declare that they have no known competing financial interest or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Keeney, R.L.; Raiffa, H. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*; Wiley: New York, NY, USA, 1976.
2. Ehrgott, M. *Multicriteria Optimization*; Springer: Berlin/Heidelberg, Germany, 2005.
3. Charnes, A.; Cooper, W.W. Programming with Goals. *Eur. J. Oper. Res.* **1961**, *1*, 39–54. [\[CrossRef\]](#)
4. Steuer, R.E. *Multiple Criteria Optimization: Theory, Computation, and Application*; Wiley: New York, NY, USA, 1986.
5. 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. [\[CrossRef\]](#)
6. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–73. [\[CrossRef\]](#)
7. Eberhart, R.; Kennedy, J. 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.
8. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [\[CrossRef\]](#)
9. Van Laarhoven, P.J.; Aarts, E.H.; van Laarhoven, P.J.; Aarts, E.H. *Simulated Annealing*; Springer: Berlin/Heidelberg, Germany, 1987.
10. 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. [\[CrossRef\]](#)
11. Zhang, Q.; Li, H. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Trans. Evol. Comput.* **2007**, *11*, 712–731. [\[CrossRef\]](#)
12. Zitzler, E.; Laumanns, M.; Thiele, L. SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK Rep.* **2001**, *103*. Available online: <https://neo.lcc.uma.es/emoo/zitzler01.ps.gz> (accessed on 23 September 2024).
13. Coello, C.C.; Lechuga, M.S. MOPSO: A proposal for multiple objective particle swarm optimization. In Proceedings of the 2002 Congress on Evolutionary Computation, CEC’02 (Cat. No. 02TH8600), Honolulu, HI, USA, 12–17 May 2002; Volume 2, pp. 1051–1056.
14. Smith, K.I.; Everson, R.M.; Fieldsend, J.E. Dominance measures for multi-objective simulated annealing. In Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753), Portland, OR, USA, 19–23 June 2004; Volume 1, pp. 23–30.
15. Mirjalili, S.; Jangir, P.; Saremi, S. Multi-objective ant lion optimizer: A multi-objective optimization algorithm for solving engineering problems. *Appl. Intell.* **2017**, *46*, 79–95. [\[CrossRef\]](#)
16. Mirjalili, S.; Jangir, P.; Mirjalili, S.Z.; Saremi, S.; Trivedi, I.N. Optimization of problems with multiple objectives using the multi-verse optimization algorithm. *Knowl.-Based Syst.* **2017**, *134*, 50–71. [\[CrossRef\]](#)
17. Alaya, I.; Solnon, C.; Ghedira, K. Ant colony optimization for multi-objective optimization problems. In Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007), Patras, Greece, 29–31 October 2007; Volume 1, pp. 450–457.
18. Chen, Z.G.; Zhan, Z.H.; Lin, Y.; Gong, Y.J.; Gu, T.L.; Zhao, F.; Yuan, H.Q.; Chen, X.; Li, Q.; Zhang, J. Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach. *IEEE Trans. Cybern.* **2018**, *49*, 2912–2926. [\[CrossRef\]](#)
19. Zhou, S.Z.; Zhan, Z.H.; Chen, Z.G.; Kwong, S.; Zhang, J. A multi-objective ant colony system algorithm for airline crew rostering problem with fairness and satisfaction. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 6784–6798. [\[CrossRef\]](#)
20. Zhang, X.; Zhan, Z.H.; Fang, W.; Qian, P.; Zhang, J. Multipopulation ant colony system with knowledge-based local searches for multiobjective supply chain configuration. *IEEE Trans. Evol. Comput.* **2021**, *26*, 512–526. [\[CrossRef\]](#)
21. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [\[CrossRef\]](#)
22. Zhan, Z.H.; Li, J.; Cao, J.; Zhang, J.; Chung, H.S.H.; Shi, Y.H. Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems. *IEEE Trans. Cybern.* **2013**, *43*, 445–463. [\[CrossRef\]](#)
23. Liu, S.C.; Chen, Z.G.; Zhan, Z.H.; Jeon, S.W.; Kwong, S.; Zhang, J. Many-objective job-shop scheduling: A multiple populations for multiple objectives-based genetic algorithm approach. *IEEE Trans. Cybern.* **2021**, *53*, 1460–1474. [\[CrossRef\]](#)
24. Tejani, G.G.; Kumar, S.; Gandomi, A.H. Multi-objective heat transfer search algorithm for truss optimization. *Eng. Comput.* **2021**, *37*, 641–662. [\[CrossRef\]](#)
25. Tejani, G.G.; Pholdee, N.; Bureerat, S.; Prayogo, D. Multiobjective adaptive symbiotic organisms search for truss optimization problems. *Knowl.-Based Syst.* **2018**, *161*, 398–414. [\[CrossRef\]](#)
26. Kumar, S.; Tejani, G.G.; Pholdee, N.; Bureerat, S.; Jangir, P. Multi-objective teaching-learning-based optimization for structure optimization. *Smart Sci.* **2022**, *10*, 56–67. [\[CrossRef\]](#)
27. Khodadadi, N.; Talatahari, S.; Dadras Eslamlou, A. MOTE: A novel multi-objective thermal exchange optimization algorithm for engineering problems. *Soft Comput.* **2022**, *26*, 6659–6684. [\[CrossRef\]](#)
28. Kumar, S.; Jangir, P.; Tejani, G.G.; Premkumar, M.; Alhelou, H.H. MOPGO: A new physics-based multi-objective plasma generation optimizer for solving structural optimization problems. *IEEE Access* **2021**, *9*, 84982–85016. [\[CrossRef\]](#)
29. Khodadadi, N.; Azizi, M.; Talatahari, S.; Sareh, P. Multi-objective crystal structure algorithm (MOCryStAl): Introduction and performance evaluation. *IEEE Access* **2021**, *9*, 117795–117812. [\[CrossRef\]](#)

30. Nouri-Moghaddam, B.; Ghazanfari, M.; Fathian, M. A novel multi-objective forest optimization algorithm for wrapper feature selection. *Expert Syst. Appl.* **2021**, *175*, 114737. [\[CrossRef\]](#)
31. Yüzgeç, U.; Kusoglu, M. Multi-objective harris hawks optimizer for multiobjective optimization problems. *BSEU J. Eng. Res. Technol.* **2020**, *1*, 31–41.
32. Abdel-Basset, M.; Mohamed, R.; Mirjalili, S.; Chakraborty, R.K.; Ryan, M. An efficient marine predators algorithm for solving multi-objective optimization problems: Analysis and validations. *IEEE Access* **2021**, *9*, 42817–42844. [\[CrossRef\]](#)
33. Tawhid, M.A.; Savsani, V. Multi-objective sine-cosine algorithm (MO-SCA) for multi-objective engineering design problems. *Neural Comput. Appl.* **2019**, *31*, 915–929. [\[CrossRef\]](#)
34. Azizi, M.; Talatahari, S.; Khodadadi, N.; Sareh, P. Multiobjective atomic orbital search (MOAOS) for global and engineering design optimization. *IEEE Access* **2022**, *10*, 67727–67746. [\[CrossRef\]](#)
35. Sabarinath, P.; Thansekhar, M.; Saravanan, R. Multiobjective optimization method based on adaptive parameter harmony search algorithm. *J. Appl. Math.* **2015**, *2015*, 165601. [\[CrossRef\]](#)
36. Got, A.; Moussaoui, A.; Zouache, D. A guided population archive whale optimization algorithm for solving multiobjective optimization problems. *Expert Syst. Appl.* **2020**, *141*, 112972. [\[CrossRef\]](#)
37. Pathak, S.; Mani, A.; Sharma, M.; Chatterjee, A. Decomposition Based Quantum Inspired Salp Swarm Algorithm for Multiobjective Optimization. *IEEE Access* **2022**, *10*, 105421–105436. [\[CrossRef\]](#)
38. Shi, Y.; Liu, H.; Gao, L.; Zhang, G. Cellular particle swarm optimization. *Inf. Sci.* **2011**, *181*, 4460–4493. [\[CrossRef\]](#)
39. Lopes, R.A.; de Freitas, A.R. Island-cellular model differential evolution for large-scale global optimization. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Berlin, Germany, 15–19 July 2017; pp. 1841–1848.
40. Seck-Tuoh-Mora, J.C.; Hernandez-Romero, N.; Lagos-Eulogio, P.; Medina-Marin, J.; Zuñiga-Peña, N.S. A continuous-state cellular automata algorithm for global optimization. *Expert Syst. Appl.* **2021**, *177*, 114930. [\[CrossRef\]](#)
41. Vafashoar, R.; Morshedlou, H.; Rezvanian, A.; Meybodi, M.R.; Vafashoar, R.; Morshedlou, H.; Rezvanian, A.; Meybodi, M.R. Applications of cellular learning automata and reinforcement learning in global optimization. In *Cellular Learning Automata: Theory and Applications*; Springer: Cham, Switzerland, 2021; pp. 157–224.
42. Seck-Tuoh-Mora, J.C.; Lopez-Arias, O.; Hernandez-Romero, N.; Martínez, G.J.; Volpi-Leon, V. A New Algorithm Inspired on Reversible Elementary Cellular Automata for Global Optimization. *IEEE Access* **2022**, *10*, 112211–112229. [\[CrossRef\]](#)
43. Zhu, G.; Ma, L. Cellular ant algorithm for multi-objective function optimization. *Control Decis.* **2007**, *22*, 1317.
44. Sidiropoulos, E. Multi-objective cellular automata optimization. In Proceedings of the Cellular Automata: 10th International Conference on Cellular Automata for Research and Industry, ACRI 2012, Santorini Island, Greece, 24–27 September 2012; Proceedings 10; Springer: Berlin/Heidelberg, Germany, 2012; pp. 131–140.
45. Zhu, D.; Zhan, T.; Zhang, Y.; Tian, H. Algorithm and application of cellular multi-objective particle swarm optimization. *Trans. Chin. Soc. Agric. Mach.* **2013**, *44*, 280–287.
46. Birashk, A.; Kordestani, J.K.; Meybodi, M.R. Cellular teaching-learning-based optimization approach for dynamic multi-objective problems. *Knowl.-Based Syst.* **2018**, *141*, 148–177. [\[CrossRef\]](#)
47. Seck-Tuoh-Mora, J.C.; Hernandez-Romero, N.; Santander-Baños, F.; Volpi-Leon, V.; Medina-Marin, J.; Lagos-Eulogio, P. A majority–minority cellular automata algorithm for global optimization. *Expert Syst. Appl.* **2022**, *203*, 117379. [\[CrossRef\]](#)
48. Nematollahi, A.F.; Rahiminejad, A.; Vahidi, B. A novel multi-objective optimization algorithm based on Lightning Attachment Procedure Optimization algorithm. *Appl. Soft Comput.* **2019**, *75*, 404–427. [\[CrossRef\]](#)
49. Kokkolaras, M.; Audet, C.; Hare, W. *Derivative-Free and Blackbox Optimization*; Springer Series in Operations Research and Financial Engineering; Springer: Berlin/Heidelberg, Germany, 2017; p. 302. [\[CrossRef\]](#)
50. Nadeau, P.C. Multiobjective Nelder-Mead Algorithm Using a Mesh-Map of Weighted Sums. Ph.D. Thesis, University of British Columbia, Vancouver, BC, USA, 2020.
51. Zenil, H.; Martinez, G.J. Cellular automata. *Scholarpedia* **2024**, *19*, 53227. [\[CrossRef\]](#)
52. Clarke, K.C. Cellular automata and agent-based models. In *Handbook of Regional Science*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 1751–1766.
53. Knowles, J.D.; Corne, D.W. Approximating the nondominated front using the Pareto archived evolution strategy. *Evol. Comput.* **2000**, *8*, 149–172. [\[CrossRef\]](#)
54. 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. [\[CrossRef\]](#)
55. Ulrich, T.; Bader, J.; Zitzler, E. Integrating decision space diversity into hypervolume-based multiobjective search. In Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, Portland, OR, USA, 7–11 July 2010; pp. 455–462.
56. Bandyopadhyay, S.; Pal, S.K.; Aruna, B. Multiobjective GAs, quantitative indices, and pattern classification. *IEEE Trans. Syst. Man Cybern. Part B* **2004**, *34*, 2088–2099. [\[CrossRef\]](#) [\[PubMed\]](#)
57. 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. [\[CrossRef\]](#)
58. Cheng, F.; Li, X. Generalized center method for multiobjective engineering optimization. *Eng. Optim.* **1999**, *31*, 641–661. [\[CrossRef\]](#)

59. Nowak, L.; Knypiński, Ł.; Jedryczka, C.; Kowalski, K. Decomposition of the compromise objective function in the permanent magnet synchronous motor optimization. *COMPEL Int. J. Comput. Math. Electr. Electron. Eng.* **2015**, *34*, 496–504. [[CrossRef](#)]
60. Ray, T.; Liew, K.M. Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE Trans. Evol. Comput.* **2003**, *7*, 386–396. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.