IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 23, 2021, accepted March 12, 2021, date of publication March 17, 2021, date of current version March 23, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3066323

An Efficient Marine Predators Algorithm for Solving Multi-Objective Optimization Problems: Analysis and Validations

MOHAMED ABDEL-BASSET[1], (Senior Member, IEEE), REDA MOHAMED[1], SEYEDALI MIRJALILI[2], (Senior Member, IEEE), RIPON K. CHAKRABORTTY[3], (Member, IEEE), AND MICHAEL RYAN[3], (Senior Member, IEEE)

[1]Faculty of Computers and Informatics, Zagazig University, Zagazig 44519, Egypt
[2]Centre For Artificial Intelligence Research and Optimization, Torrens University Australia, Fortitude Valley, QLD 4006, Australia
[3]Capability Systems Centre, School of Engineering and Information Technology, UNSW Canberra, Campbell, ACT 2612, Australia

Corresponding author: Reda Mohamed (redamoh@zu.edu.eg)

ABSTRACT Recently, a new strong optimization algorithm called marine predators algorithm (MPA) has been proposed for tackling the single-objective optimization problems and could dramatically fulfill good outcomes in comparison to the other compared algorithms. Those dramatic outcomes, in addition to our recently-proposed strategies for helping meta-heuristic algorithms in fulfilling better outcomes for the multi-objective optimization problems, motivate us to make a comprehensive study to see the performance of MPA alone and with those strategies for those optimization problems. Specifically, This paper proposes four variants of the marine predators' algorithm (MPA) for solving multi-objective optimization problems. The first version, called the multi-objective marine predators' algorithm (MMPA) is based on the behavior of marine predators in finding their prey. In the second version, a novel strategy called dominance strategy-based exploration-exploitation (DSEE) recently-proposed is effectively incorporated with MMPA to relate the exploration and exploitation phase of MPA to the dominance of the solutions—this version is called M-MMPA. DSEE counts the number of dominated solutions for each solution—the solutions with high dominance undergo an exploitation phase; the others with small dominance undergo the exploration phase. The third version integrates M-MMPA with a novel strategy called Gaussian-based mutation, which uses the Gaussian distribution-based exploration and exploitation strategy to search for the optimal solution. The fourth version uses the Nelder-Mead simplex method with M-MMPA (M-MMPA-NMM) at the start of the optimization process to construct a front of the non-dominated solutions that will help M-MMPA to find more good solutions. The effectiveness of the four versions is validated on a large set of theoretical and practical problems. For all the cases, the proposed algorithm and its variants are shown to be superior to a number of well-known multi-objective optimization algorithms.

INDEX TERMS Multi-objective optimization problem, dominance strategy-based exploration-exploitation, Gaussian-based mutation, Nelder-Mead simplex, marine predators algorithm.

I. INTRODUCTION

Recently, multi-objective optimization problems (MOP) have gained significant attention from researchers looking to assist decision-makers (DM) to make better choices than now. Unlike single objective problems (SOP), MOP does not have a single solution but have a set of solutions representing the best trade-offs among the multiple objectives, which are often in conflict. However, the presence of multiple optimal solutions might help the DMs finding the more relevant solution to their problem [1], [2]. The set of optimal solutions called the non-dominated solutions are the solutions that may degrade an objective if any one of the others can be improved. Many real-world problems are multi-objective such as task

The associate editor coordinating the review of this manuscript and approving it for publication was Ioannis Schizas.

scheduling [3], water distribution networks (WDNs) [4], wind speed forecasting [5], protein structure [6], and the traveling salesman problem [7].

In MOP, the focus is to find a set of solutions to a problem for which there are multiple objectives that must be maximized/minimized. The mathematical model of the MOP is as follows:

$$\begin{aligned} Max/Min\, f(x) &= \{f_1(x), f_2(x), f_3(x), f_4(x), \ldots, \\ &\quad f_m(x)\}, m \geq 2 \\ subject\ to\ nq_i(x) &\geq 0, i = 1, 2, \ldots, z \\ q_i(x) &= 0, i = 1, 2, \ldots, k \\ L_i &\leq x_i \leq U_i, i = 1, 2, \ldots, p \end{aligned} \tag{1}$$

where m is the number of objectives. nq_i determines the i^{th} inequality constraint, z determines the number of inequality constraints , q_i represents the i^{th} equality constraint, k determines the number of equality constraints, p is the number of variables, and L_i and U_i are the lower and upper bounds for the i^{th} variable (x_i).

Priori and posteriori [2], [8] approaches have been proposed for solving MOP. In a priori approach, the MOP is converted into a SOP using a weights vector that specifies the significance of each objective, and then the SOP is solved for an optimal solution. Such approaches are not normally practicable due to the need to have the DM provide a weight for each objective. As a result, the posteriori approach extracts the set of non-dominated solutions (or at least a well-distributed set of solutions) from which the DM can select an appropriate solution. Although the DM has a range of optimal solutions available, they may find difficulty in extracting an appropriate solution when there is a number to choose among. Due to their success in solving many real-world problems, meta-heuristics [9]–[12] and evolutionary algorithms [13]–[16] have been applied to MOP in many works briefly reviewed in the following paragraphs, beginning with evolutionary algorithms (EA).

Zapotecas-Martínez [17] suggested a Lebesgue indicator-based evolutionary algorithm (LIBEA) to maximize the Hybervolume of the non-dominated solutions extracted during the optimization process to reduce the high computational time with an increasing number of objectives result of using Hybervolume with EA as an indicator of solutions towards the true Pareto optimal solutions. LIBEA solved the continuous MOPs with complicated properties using the regularity property of those MOPs.

Liu and Wang [18] also proposed EA as the first attempt to solve constrained MOP simultaneously taking into consideration the constraints of both objective and decision spaces. Chen *et al.* [19] integrated EA with two methods: reference vector adaptation for tackling the problem of the different shapes of Pareto front, and diversity ranking to manage diversity. Based on the decomposition strategy, Zhang [20] proposed the EA which breaks the problems into various sub-problems, and then each is solved based on the information from the neighboring sub-problems. Additionally, Tian, Y. *et al.* [21] used an improved inverted generational distance indicator to address the different shapes of Pareto fronts using a multi-objective EA.

Seifollahi-Aghmiuni and Haddad [22] introduced a comprehensive EA to optimize SOPs and MOP by applying a unique structure. Although the set of the parameters involved didn't need more sensitivity analysis, the approach consumes considerable time. Pedroso *et al.* [23] introduced a parallel EA using differential and genetic evolution operators. To reduce the high computational time result of using the hypervolume as a performance measure to guide the search during the optimization process, Gómez *et al.* [24] provided a parallel EA based on an asynchronous separated approach with micro-populations. However, the performance of this algorithm (parallel EA) was only investigated on the walking-fishing group (WFG) test suite, in relative to the other benchmarks such as CEC [25], [26], and GLT [27] its performance is not known. The Pareto archived evolution strategy (PAES) [28] has been proposed as a local search method, to maintain an archive of the obtained non-dominated solutions so far, for identifying the dominance between the current solutions and the archiving solution.

Yen and Lu [29] proposed an EA approach based on the dynamic population size for solving MOPs. This algorithm was not able to achieve all the challenging characteristics of the test functions used to check their performance. Furthermore, differential evolution (DE) [30] has been developed for solving multi-modal MOPs. Further, it needs an improvement to increase the coverage of the solutions on the objective space.

In terms of meta-heuristic algorithms, a parallel particle swarm optimization (PSO) [31] was incorporated with a cooperative co-evolutionary approach and a strategy to control the velocity of the particles to get rid of erratic movements. Mokarram and Banan [32] also used PSO to overcome MOPs based on the selection of the leader to guide the particles in the search space to increase the diversity and convergence towards the optimal solution. Moreover, as an attempt to guide the other particles within their flight, PSO [33] was integrated with an alternative repository of particles. Mousa *et al.* [1] have been proposed PSO for solving MOPs based on two features: (1) initializing the particles randomly to fly through the search space, (2) using a local search method with PSO to search through the less-crowded area for more non-dominated solutions. Furthermore, the niching PSO [34] has been combined with a local search method to enhance the local search ability when solving the multi-model MOPs. Zhang *et al.* [35] used ring topology and an updating strategy for the leader with a novel cluster-based approach with PSO to solve the multimodal MOPs.

The crow search algorithm (CSA) [36] has been developed for tackling MOPs., Nobahari and Bighashdel [37] proposed a multi-objective CSA that used an objective function based on the weight vectors, in addition to suggesting a chasing strategy for faster convergence toward the optimal solutions. In [38], The CSA and the sine cosine algorithm (SCA)

cooperated to tradeoff between the exploration and the exploitation operators. Furthermore, to preserve the diversity among the solutions, the SCA [39] integrated by the crowding distance approach has been suggested for tackling multi-model MOP.

The performance of the salp swarm algorithm (SSA) [40] was enhanced with DE for multi-objective big-data optimization. In the same context, SSA [41] was integrated with PAES to solve MOPs. In [42], a new multi-objective optimization algorithm called evolutionary multi-objective seagull optimization algorithm (EMoSOA) has been recently proposed. This algorithm used a grid mechanism, leader selection, dynamic archive concept, and genetic operators to cache the non-dominated solutions, in addition to employing the roulette wheel method to select a solution from the archived ones. EMoSOA was extensively compared with a number of the optimization algorithms on 24 benchmark functions to observe its effectiveness.

Jiang, S. and Z. Chen [43] proposed a two-phase evolutionary algorithm framework for tackling the multi-objective optimization problems; in the first stage, it adopted a specific set of the multi-objective evolutionary algorithms with a small population size to accelerate the convergence speed toward the true Pareto optimal solutions; in the second one, a selection mechanism based on measure function and crowdedness function has been employed to improve the population's uniformity in the objective space. The performance of this framework was observed on problems with conflicting objectives up to 10; this observation showed its effectiveness in comparison to several multi-objective evolutionary algorithms.

Tian, Y., *et al.* [44] proposed an evolutionary algorithm for tackling the large-scale multimodal multiobjective optimization problems with spare Pareto optimal solutions. This algorithm used multiple subpopulations to explore the decision space as possible, in addition to using a guiding vector to guide the search behavior of the subpopulations. In [45], a novel multiobjective evolutionary algorithm was proposed with incorporating the concept of sum of objectives in the adaptive mating and environment selection mechanisms to preserve the diversity among the individuals in addition to fast move for converging to the true Pareto. This algorithm was validated on 26 test functions, and compared with some state-of-the-art methods to see its effectiveness.

Wang [46] proposed a multiobjective evolutionary algorithm integrated with a uniformly evolving scheme to produce non-dominated solutions uniformly distributed on the true Pareto optimal curve to give the decision-makers the flexibility in selecting the satisfactory solutions. Many other multi-objective algorithms are designed for dealing with MOPs such as flowers pollination algorithm [47], symbiotic organism search [48], bat algorithm [49], ant lion optimizer [50], whale algorithm [51], artificial sheep algorithm [52], immune algorithm [53], moth flame algorithm [54], [55], grey wolf optimizer [56], grasshopper algorithm [57], [58], multi-objective equilibrium optimizer [59], multi-objective whale optimization algorithm [60], and several else [61]–[66].

Recently, a new meta-heuristic optimization algorithm called the marine predators algorithm (MPA) has been proposed for solving continuous optimization problems [67]. MPA simulates the behavior of predators in attacking their prey and uses Brownian and Lévy steps as the means of predators searching for their prey, in which the velocity ratio from the prey to the predators is used to tradeoff between those two strategies. Besides, due to surrounding environmental issues, the predators spend 80% of their time searching for their prey in the vicinity, while the remaining time is spent in other environments. This process is called fish aggregating devices (FADs) and gives the algorithm a powerful ability to avoid being trapped in local optima. Despite its demonstrated advantages and significant successes for several optimization problems [68], [69], MPA has not been applied to the solution of MOPs–consequently, this paper proposes the multi-objective version of MPA.

In this paper, four new models have been proposed for helping the DMs by taking the right decision when solving a MOP. Before giving a short outline of each proposed model, our methodology is first illustrated. Our methodology is based on proposing new strategies to assist all meta-heuristic algorithms and especially the MPA when solving the MOPs. The first strategy is the dominance strategy based exploration-exploitation (DSEE), in which the dominated solutions by each solution are counted and the solution with a high dominated count will be exploited and the others will undergo the exploration operator. This strategy tries to help any meta-heuristic algorithm when solving MOPs by relating the exploration and exploitation phases of the algorithm with the dominance strategy instead of the iterations.

Besides the Gaussian-based mutation (GM) strategy, based on exploration and exploitation, is proposed to help the optimization algorithm to reach more good solutions. GM tries to mutate the current solution with a probability to swap between moving towards the upper bound or the lower bound with a small step size (exploitation) or a large step size as an attempt to find more solutions in the vicinity based on the Gaussian distribution. Because most of the meta-heuristic algorithms are based on moving toward the best solutions (non-dominated solutions) found so-far, the Nelder-Mead method (NMM) has been suggested as the third strategy. Executing this strategy at the start of the optimization process will build up a front of the non-dominated solutions that will help all meta-heuristic algorithms in reaching better solutions.

In this work, the first proposed model is called the multi-objective marine predators' algorithm (MMPA)—this model adapts the standard MPA for addressing MOPs. To improve the performance of the MMPA, the search methodology is modified using DSEE to relate its exploration and exploitation phases with the dominance strategy. This model is called the multi-objective modified marine predators algorithm (M-MMPA). The third model, abbreviated as M-MMPA-GM,

integrates the M-MMPA with the GM to assist in reaching the optimal solution. In the fourth model, abbreviated as M-MMPA-NMM, the NMM is applied to a number of the non-dominated solutions selected randomly at the start of the optimization process to construct a front of the solutions that will help M-MMPA in reaching better solutions within less time possible.

The main contributions of this paper are summarized as follows:

1) Proposing three versions of MPA, in addition to the standard version for solving MOP.
2) Adapting two recently-published strategies namely dominance strategy-based exploration-exploitation, and Gaussian-based strategy with MPA to improve its performance on MOP.
3) Integrating the MPA with the Nelder-Mead simplex method to promote its searching ability for reaching more non-dominated solutions.
4) The results indicate that our proposed algorithms perform better or on par with NSGA-II, NSGA-III, CGD, GDE3, and SMPSO algorithms.

This paper is organized as follows. Section 2 briefly describes MPA. Section 3 describes the proposed algorithms for tackling MOPs. Section 4 illustrates the experimental settings. Section 5 provides the discussion and the experimental results comparing the proposed algorithms using the CEC 2020, CEC 2009, and GLT test functions. Section 6 provides some conclusions regarding the proposed approach and future work.

II. MARINE PREDATORS ALGORITHM (MPA)

Recently, Faramarzi *et al.* [70] proposed MPA to mimic the behavior of marine predators in search of their prey, in which the predators use the velocity ratio between the predators and prey for the tradeoff between Brownian, and Lévy strategies, which are considered the optimal strategies for the predator to find their prey. The mathematical model of MPA is formulated as:

At the initialization step, a number of the prey is distributed within the search space of the optimization problem using the following equation:

$$\vec{X} = \vec{X}_{min} + \vec{r} * (\vec{X}_{max} - \vec{X}_{min}) \tag{2}$$

where $\vec{r}$ is a vector generated randomly within 0 and 1, and $\vec{X}_{min}$, $\vec{X}_{max}$ are vectors containing the maximum and the minimum of the boundaries of the optimization problems.

Each distributed solution is then evaluated using the objective function and the highest solution having the highest objective value is used as the top predator within the optimization process. Based on the survival of the fitness theorem, the top predator is used to build up a matrix known as Elite, and abbreviated as E. This elite matrix is shown as follows:

$$E = \begin{bmatrix} A^I_{1,1} & A^I_{1,2} & \dots & A^I_{1,d} \\ A^I_{2,1} & A^I_{2,2} & \dots & A^I_{2,d} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ A^I_{N,1} & A^I_{N,2} & \dots & A^I_{N,d} \end{bmatrix}$$

where $\vec{A^I}$ represents the top predator vector and is repeated N times to construct E., n is the number of individuals within the population, and d is the number of dimensions.

The predators will be updated towards another matrix called as *prey*, and abbreviated as *py*, this matrix is formulated as follows:

$$py = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,d} \\ A_{2,1} & A_{2,2} & \dots & A_{2,d} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ A_{N,1} & A_{N,2} & \dots & A_{N,d} \end{bmatrix}$$

In the main loops of the MPA, the optimization process will be divided into three stages based on the velocity-ratio, illustrated previously, and is modeled as follows:

A. HIGH-VELOCITY RATIO

In this phase, the predators will wait to watch the motion of the prey that moves quickly to search for its food. Generally, this phase is called the exploration phase, where the prey starts to discover the search space using the Brownian strategy for finding the promising regions that may include the optimal solution. Accordingly, the location of the current prey is updated within the first third of the maximum iterations according to the following:

$$if\ t < \frac{1}{3} t_{max}$$

$$\vec{S}_i = \vec{R}_B \otimes \left(\vec{E}_i - \vec{R}_B \otimes \vec{Py}_i\right) \tag{3}$$

$$\vec{py}_i = \vec{py}_i + P.\vec{R} \otimes \vec{S}_i \tag{4}$$

where $\vec{R}_B$ is a numerical vector created randomly based on the normal distribution, $\otimes$ represents the entry-wise multiplication, P is a fixed numeral (0.5 is recommended), $\vec{R}$ is a numerical vector generated uniformly, . indicates the multiplication operator, t is the current iteration, and t_{max} is the maximum number of iterations.

B. UNIT VELOCITY RATIO

After exploring the search space within the first third of the maximum iteration, the promising regions that may include the optimal solution are so-near; therefore, this phase is considered as the transition stage between the exploration and the exploitation. Generally, in this phase, the exploration phase will be gradually converted into the exploitation phase, so this is an intermediate phase between the exploration and exploitation operators. Specifically, in this phase, the predators are responsible for the exploration, while the prey

for exploitation. For representing this phase using a mathematical model, half of the population would be moved using the exploration operator (Brownian motion), and the other half will be moved using exploitation operator (Levy flight), as illustrated in the following equations:

$$if\ \frac{1}{3}t_{max} < t < \frac{2}{3}t_{max}$$

- For the first half of the population

$$\vec{S}_i = \vec{R}_L \otimes \left(\vec{E}_i - \vec{R}_L \otimes \vec{P}y_i\right) \tag{5}$$

$$\vec{py}_i = \vec{py}_i + P.\vec{R} \otimes \vec{S}_i \tag{6}$$

- For the second half of the population

$$\vec{S}_i = \vec{R}_B \otimes \left(\vec{R}_B \otimes \vec{E}_i - \vec{Py}_i\right) \tag{7}$$

$$\vec{py}_i = \vec{py}_i + P.CF \otimes \vec{S}_i \tag{8}$$

where CF is an adaptive parameter used to monitor the step size and is formulated as follows:

$$CF = (1 - \frac{t}{t_{max}})^{\left(2\frac{t}{t_{max}}\right)} \tag{9}$$

C. LOW-VELOCITY RATIO

In the last stage, the prey location is already detected and the predators hurry to catch them. This phase is executed in the last third of the maximum iteration, where the predators follow the levy step to update its position using the following equation:

$$if\ t > \frac{2}{3}t_{max}$$

$$\vec{S}_i = \vec{R}_L \otimes \left(\vec{R}_L \otimes \vec{E}_i - \vec{Py}_i\right) \tag{10}$$

$$\vec{py}_i = \vec{py}_i + P.CF \otimes \vec{S}_i \tag{11}$$

Another advantage of MPA is that it mimics predators' behaviors to increase the probability of escaping from local optima. This advantage is that the surrounding environment effects such as eddy formulation and fish aggregating devices *(FADs)* would affect the behavior of the predators. As a result, 20% of the time, the predators would jump into other regions with abundant prey, while the remainder of the time, they search for their prey in the local vicinity. FADs can be formulated as follows:

$$\vec{py}_i = \begin{cases} \vec{py}_i + CF[X_{min} + \vec{R} \otimes (\vec{X}_{max} - \vec{X}_{min})] \otimes \vec{U} \\ \quad if\ r < FADS \\ \vec{py}_i + [FADS\,(1-r) + r]\left(\vec{py}_{r1} - \vec{py}_{r2}\right) \\ \quad if\ r \geq FADS \end{cases} \tag{12}$$

where r is a random number in the range of [0,1]. $FADs = 0.2$ refers to the impact of the $FADs$ on the optimization process. $\vec{U}$ is a binary vector containing 0 and 1 values, which are assigned based on creating a random number vector equal to the size of $\vec{U}$ and assigning it with random numbers between 0 and 1. The values inside this vector that are smaller than the FADs' value is assigned a value of 1 in $\vec{U}$, otherwise, they are assigned 0. $r1$, and $r2$ are numerical values randomly generated within the range of the population size.

MPA accomplishes memory saving by saving the old position of the prey. After updating the current solutions, the fitness values of each current solution and each old solution are compared, and if the fitness of the old one is better than the current one, they are swapped. The steps of MPA are listed in Algorithm 1.

Algorithm 1 The Marine Predator Algorithm (MPA)

```
Initialize py, P = 0.5
while (t < t_max)
  Calculate the fitness value for each py_i|i = 0, 1, 2, 3, ....., N
  if (t == 0)
    Construct E matrix
  Else
    Update E, if there is better
  end if
  Assign CF using Eq. (9)
  for each i py
    Update the current py_i using one of the following equations Eq. (4), (5 or 7), and (11)
  end for
  Calculate the fitness value of py
  Update E, if there is better
  Accomplish the memory saving
  implement the FADS using Eq. (12)
  t++
end while
```

III. MULTI-OBJECTIVE MARINE PREDATORS ALGORITHMS VERSIONS

This section proposes four multi-objective variants of MPA. Our proposed versions will use an external archive to handle the non-dominated solutions found within the optimization process. In addition, the crowded distance strategy has been used to preserve the diversity among the non-dominated solutions.

A. INITIALIZATION

In this phase, a N prey is generated, and each prey has n variables at the expense of the size of MOP. Those variables are initialized using the following formula:

$$\forall i \in \mathrm{N}, py_{i,j} = XL_j + r * (XU_j - XU_j) \tag{13}$$

where XU_j, and XL_j are the upper and lower bound of variable j, $py_{i,j}$ refers to the j^{th} variable of the i^{th} prey and r is a random number in the range [0, 1]. Finally, the steps of the multi-objective version for the standard MPA with the archive is illustrated in Algorithm 2. This standard version is called multi-objective marine predators algorithm (MMPA).

Algorithm 2 The Multi-Objective Marine Predator Algorithm (MMPA)

1. Initialize py, $P = 0.5$
2. Create an archive, namely *archive*, containing the non-dominated solutions within the optimization process
3. **while** $(t < t_{max})$
4. Calculate the fitness value of py
5. Update *archive*
6. Initialize E matrix from *archive* randomly
7. Assign CF using Eq. (9)
8. **for** each i prey
9. Based on the current iteration, Update the current py_i using one of the following equations Eq. 3, (5 or 7), and 10
10. **end for**
11. Calculate the fitness value of **py**
12. Update *archive*
13. Accomplish the memory saving
14. implement the *FADS* using Eq. (12)
15. $t++$
16. end while

B. DOMINANCE STRATEGY BASED EXPLORATION AND EXPLOITATION (DSEE)

Before demonstrating the modifications on MPA for solving MOP, this section briefly describes how meta-heuristics algorithms search for the optimal solution. Meta-heuristic algorithms first explore the search space to identify the most promising region. The exploration then gradually reduces until it is converted into the exploitation phase, which focuses on the optimal solution found so far. If the optimal solution found so far is a local optima solution, the algorithm will focus on that solution and not be able to escape to reach a global optimum due to fading away from the diversity of the solutions that ensure exploring more regions to reach better solutions. So, a new methodology is proposed in [71] to avoid being trapped in local optima when solving MOP and this methodology is consist of the following steps:

1) Calculate the number of solutions dominated by each solution [72].
2) The solution with the highest number of dominated solutions will be used for the exploitation phase, due to its probability of being near to the true Pareto optimal solution.
3) The solution with the smallest number of (or no) dominated solutions will have an exploration phase to explore another region seeking a better solution.

After calculating the dominated solutions count for each solution in the population using Algorithm 3 and storing it into $\vec{R}$, the new implementation of MPA is as follows:

Eq. (4) is implemented if R_i of the i^{th} prey is smaller than LC (predefined by users), so Eq. (4) is reformulated as follows:

Algorithm 3 Ranking Solution Based on Dominated Solutions [72]

1. R : a vector of size N initializing with 0's value
2. $i = 0$
3. **while** $(i < N)$
4. $j = 0$
5. **While**$(j < N)$
6. **If** $(i \mathrel{!}= j)$
7. **If** $(py_i dominated py_j)$
8. R_i++
9. **End if**
10. **end if**
11. **end while**
12. $i++$
13. **end while**
14. **return** R

$$if\ R_i < LC$$
$$\vec{S}_i = \vec{R}_B \otimes \left(\vec{E}_i - \vec{R}_B \otimes \vec{Py}_i\right) \tag{14}$$
$$\vec{py}_i = \vec{py}_i + P.\vec{R} \otimes \vec{S}_i \tag{15}$$

where LC indicates the maximum value of dominated solutions count used to determine if the i^{th} solution will be updated using Eq. (15) or not. If R_i of the i^{th} solution is lower than LC, then this solution is updated using Eq. (15).

Also, Eq. (11) is implemented if R_i of the i^{th} prey is greater than UC (predefined by users), so Eq. (11) will be reformulated as follows:

$$if\ R_i > UC$$
$$\vec{S}_i = \vec{R}_L \otimes \left(\vec{R}_L \otimes \vec{E}_i - \vec{Py}_i\right) \tag{16}$$
$$\vec{py}_i = \vec{py}_i + P.\vec{R} \otimes \vec{S}_i \tag{17}$$

where UC indicates the minimum value of dominated solutions count used to determine if the i^{th} solution will be updated using Eq. (17) or not. Eq. (6), and (8) are also reformulated as follows:

$$if\ LC < R_i < UC$$

- For the first half of the population

$$\vec{S}_i = \vec{R}_L \otimes \left(\vec{E}_i - \vec{R}_L \otimes \vec{Py}_i\right) \tag{18}$$
$$\vec{py}_i = \vec{py}_i + P.\vec{R} \otimes \vec{S}_i \tag{19}$$

- For the second half of the population

$$\vec{S}_i = \vec{R}_B \otimes \left(\vec{R}_B \otimes \vec{E}_i - \vec{Py}_i\right) \tag{20}$$
$$\vec{py}_i = \vec{py}_i + P.CF \otimes \vec{S}_i \tag{21}$$

The steps of the multi-objective modified MMPA using DSEE are elaborated in Algorithm 4.

Algorithm 4 The Multi-Objective Modified Marine Predator Algorithm (M-MMPA)

1. Initialize py, P = 0.5
2. Create an archive, namely *archive*, containing the non-dominated solutions within the optimization process
3. **while** (t < t_{max})
4. Calculate the fitness value for each $py_i|i = 0, 1, 2, 3, \ldots\ldots, N$
5. Update *archive*
6. Initialize E matrix from *archive* randomly
7. Assign CF using Eq. (9)
8. **for** each i prey
9. Based on the value of R_i, Update the current py_i using one of the following equations Eq. 14, (18 or 20), and 16
10. **end for**
11. Calculate the fitness value of *py*
12. Update *archive*
13. Accomplish the memory saving
14. implement the FADS using Eq. (12)
15. t++
16. end while

Algorithm 5 ASF Algorithm

1. *Max_Ratio* : Negative_Infinity
2. W : a vector of size equal to the number of objectives and contain random values their sum equal 1
3. f : a vector contains the objective function values of the current solution
4. $i = 0$
5. **while** ($i < N$)
6. $Max\_Ratio = Max(Max\_Ratio, \frac{f_i}{w_i})$
7. $i++$
8. **end while**
9. **return** *Max_Ratio*

C. ACHIEVEMENT SCALARIZATION FUNCTION (ASF)

The ASF [73] is used to convert MOP into SOP using a predefined weight vector containing several weights equal to the number of objectives of the MOP, where the weights express the significance of each objective. ASF can produce weakly Pareto optimal solutions. The mathematical model of ASF as follows:

$$u^{asf}(f; w) = max\left(\frac{f_i}{w_i}\right) \tag{22}$$

where f_i is the value of the objective i^{th} and w_i is the weight of the same objective i^{th}. Algorithm 5, illustrates the steps of converting a MOP with N objectives into a single objective (ASF steps) [73].

D. NELDER-MEAD METHOD (NMM)

NMM is a method used to extract the maximum or the minimum of an objective function in a multidimensional space. The NMM algorithm is listed as follows:

1) INITIAL SIMPLEX

At the start, a simplex consisting of $n + 1$ points $P_0, P_1, P_2 \ldots\ldots\ldots\ldots, P_n$ around the input point P_{in} is constructed. Those points are generated around the input as follows:

1) $P_0 = P_{in}$
2) $P_i = P_0 + h_i.e_i, j = 1, \ldots, n$

Where h_i is a stepsize in the same direction of the unit vector e_i

2) MAIN STEPS

Each iteration, the NMM implements the steps given in Algorithm 6.

For more information regarding NMM, see [74]. In addition, Algorithm 7 illustrates the steps of integrating the modified version of MPA with NMM (M-MMPA-NMM). NMM is implemented within the first MIter iterations to construct a front of the non-dominated solutions that will help M-MMPA to reach better solutions within the optimization process. Because NMM only deals with SOP, ASF has been used to convert the multiple objectives into a single objective. In each iteration of the first MIt*er*, *NSS* non-dominated solutions will be selected randomly from the archive, and then updated using NMM and, if better, added again into the archive.

E. GAUSSIAN-BASED MUTATION

In this section, the Gaussian mutation proposed in [71] is used as a tool to increase the convergence towards the true Pareto optimum by searching widely around the current solution (exploration) and narrowly (exploitation). This will help in accelerating the convergence toward a better non-dominated solution in the case of finding one near the current one. On the other hand, if this current solution is local minima, the exploration capability will update it to a long-distance up to the upper bound of each variable and down to the lower bound, and subsequently falling into local minima is substantially avoided. The mathematical model of the Gaussian equation is formulated as follows:

$$g = \sigma * \sqrt{(-2.0\, log\,(r_1))} * sin\,(2.0\pi\, r_2) \tag{23}$$

where σ is the sigma value and is between 0 and 1, r_1, and r_2 are random numbers at the range of 0 and 1. The steps of Gaussian-based mutation are illustrated in Algorithm 8. In Algorithm 8, the GM is applied a number of times Nt, predefined by users, on the solution X. According to the mutation probability (MP), for each time out of Nt, a number, r_4, is generated randomly at the range of [0,1], and if r_4 is greater than 0.5, then the updating of the i^{th} variable of the solution X will be towards the difference between the

Algorithm 6 The Main Steps of NMM

1. Calculate the value of each point P_i and store this value into y_i.
2. Define h as the suffix such that $y_h = max\,(y_i)$, and l as the suffix such that $y_l = min\,(y_i)$.
3. Define the centroid of the points $P_i|i \neq h$ and call it $\bar{P}$.
4. Compute the reflection point P^* using the following equation :

$$P^* = (1+\alpha)\,\bar{P} - \alpha P_h$$

where α is the reflection constant.

5. Compute y^* of the P^*.
6. **If** $(y^* > y_l$ *and* $y^* < y_h)$:

$$P_h = P^*$$

7. **If** $(y^* < y_l)$ **Then:**
P^* are expanded to P^{**} using the following formula:

$$P^{**} = \gamma\bar{P} + (1-\gamma)\,P_h$$

where $\gamma > 1$ is the expansion coefficient

8. Compute y^{**} of the P^{**}
9. **If** $(y^{**} < y_l)$ **Then:**
$P_h = P^{**}$ and restart the process
10. **If** $(y^{**} > y_l)$ **Then**
P_h is replaced with P^* before restarting.
11. **If** $(y^* > y_i \forall i \neq h)$ **Then:**
P_h are contracted to P^{**} using the following formula:

$$P^{**} = \beta\bar{P} + (1-\beta)\,P_h$$

where β is the contraction coefficient and lies between 0 and 1.

12. Compute y^{**} of the P^{**}
13. **if** $(y^{**} < y_h)$:
$P_h = P^{**}$ and restart the process
Else:
$\forall i P_i = ((P_i + P_l)/2)$ and restart the process

upper bound (XU_i) and the current X_i values of the same variable otherwise it will be moved toward the difference between the current X_i and the lower bound (XL_i). 0.5 is the recommended value. The calculated difference either toward the upper bound or the lower bound is stored into *Del* variable. After that, a new random number r_5 is generated to tradeoff between the exploration and the exploitation phase. If r_5 is lower than the exploitation probability (EP) ($r_5 < EP$), then the i^{th} variable is updated according to lines 19 and 20 in Algorithm 8, otherwise is updated according to lines 22 and 23. σ is recommended for the exploitation with a value between 0 and 0.3, and for the exploration with a value between 0.3 and 1.0.

Finally, Algorithm 9 illustrates the steps of integrating the modified version of MPA with Gaussian-based mutation (M-MMPA-GM) for tackling MOP. Gaussian-based mutation

Algorithm 7 The Multi-Objective Modified Marine Predator Algorithm With NMM (M-MMPA-NMM)

1. Initialize py, $P = 0.5$
2. Create an archive, namely *archive*, containing the non-dominated solutions within the optimization process
3. **while** $(t < t_{max})$
4. Calculate the fitness value for each $py_i|i = 0, 1, 2, 3, \ldots\ldots, N$
5. Update *archive*
6. Initialize E matrix from *archive* randomly
7. Assign CF using Eq. (9)
8. **for** each i prey
9. Based on the value of R_i, Update the current py_i using one of the following equations Eq. (15), (18 or 20), and (17)
10. **end for**
11. Calculate the fitness value of py
12. Update *archive*
13. Accomplish the memory saving
14. implement the FADS using Eq. (12)
15. Within *minimum iterations(MIter)*, Calling NMM to update a randomly selected solution *NSS* from the *archive*
16. t++
17. end while

works on searching around the current solution widely (exploration) and narrowly (exploitation) finding more good solutions in fewer iterations, and also helps to reach solutions that M-MMPA cannot reach. In each iteration, *NSS* of the non-dominated solutions will be selected randomly from the archive, and each one will be updated a number *Nt* of times using Gaussian-based mutation and, if a better solution, added again into the archive.

IV. EXPERIMENTAL SETTINGS

In this section, the parameter settings, performance metrics, and multi-objective test functions are described in detail.

A. MULTI-OBJECTIVE TEST FUNCTIONS

CEC 2020, CEC 2009, and GLT benchmark problems are used to validate the performance of our proposed algorithms statistically compared with some state of art algorithms such as non-dominated sorting genetic algorithm II (NSGA-II) [75], non-dominated sorting genetic algorithm III (NSGA-III) [76], speed-constrained multi-objective particle swarm optimization (SMPSO) [77], generalized differential evolution (GDE3) [78], and constrained decomposition with grids for evolutionary multi-objective optimization (CDG) [79]. These datasets contain the most recent and challenging test functions with different shapes for Pareto optimal fronts such as convex, non-convex, and disconnected. Sixteen test functions are selected from CEC 2020. CEC 2009 consists of 10 test functions having two or three objectives and

Algorithm 8 Gaussian-Based Mutation (GM)

Input: is a vector containing the solution to mutate, namely X
Output: the mutated X
MP : the mutation probability
$Y = X$
$i = 0$
EP: the exploitation probability
$j = 0$
While ($j < Nt$)
  while ($i<n$)
    r_3 *is a number generated randomly within 0 and 1*
    If ($r_3 < MP$)
      r_4 *is a number generated randomly within 0 and 1*
      If ($r_4 \leq 0.5$)
        $Del = X_i - XL_i$
      Else
        $Del = XU_i - X_i$
      End If
      If ($r_5 <$ EP)
        Calculate g using Eq. (26) with σ value between 0 and 0.3
          $X_i = X_i + g * Del$
      Else
        Calculate g using Eq. (26) with σ value between 0.3 and 1.0
          $X_i = X_i + g * Del$
      End If
    End if
      $i++$
    end while
    If *(X dominated Y)*
      $Y = X$
    Else
      $X = Y$
  j++
  end while

Algorithm 9 The Multi-Objective Modified Marine Predator Algorithm With Gaussian (M-MMPA-GM)

Initialize py, $P = 0.5$
Create an archive, namely *archive*, containing the non-dominated solutions within the optimization process
while ($t < t_{max}$)
  Calculate the fitness value for each $py_i | i = 0, 1, 2, 3, \ldots\ldots, N$
  Update *archive*
  Initialize E matrix from *archive* randomly
  Assign CF using Eq. (9)
  for each i prey
    Based on the value of R_i, Update the current py_i using one of the following equations Eq. (15), (18 or 20), and (17)
  end for
  Calculate the fitness value of py
  Update *archive*
  Accomplish the memory saving
  implement the *FADS* using Eq. (12)
  Calling Algorithm 5 to update a number *NSS* of solutions selected randomly from the *archive*
  $t++$
end while

thirty variables and GLT problems consist of six test functions with two or three objectives. All GLT problems have ten variables.

The CEC 2020 problem set is considered to be multi-modal multi-objective because it includes problems with a point on the Pareto front in the objective space that can be obtained by more than one Pareto optimal solution in the decision space. However, CEC 2009 and GLT problems are only are multi-objective since each point on the Pareto front in the objective space corresponds to only one point on the optimal Pareto solution. Fig.1 shows an example of multi-modal multi-objective—point P1 in the objective space can be obtained by either S2 or S3 Pareto solutions in the decision space. Similarly, point P2 can also be obtained by two different Pareto solutions S1 and S4 in the decision space. Table 1 provides a description of the three benchmark problem sets. We used a device with Intel® Core™ i3-2330M CPU @ 2.20 GHz, 1GB of RAM, and prepared with Windows 7 Ultimate platform. Further, All algorithms in our experiments were compared with the proposed algorithms according to their implementation in the JMetal framework [80] based on the parameters found in the published papers.

B. SENSITIVITY ANALYSIS

One of the most important factors that affect the performance of the meta-heuristic algorithms is the parameter values selected for each algorithm. For extracting the best parameters for the proposed versions, the set of GLT test problems is used to test the performance of the algorithms with different parameters values—the results are given in Table 2, from which it can be noted that the values 0 and 40 for both *LC* and *UC,* respectively, of the proposed algorithm: M-MMPA are better than all the other values. Table 3 contains the results for the parameters of M-MMPA-NMM—from which it can be noted that when the number of randomly selected solutions, *NSS*, and the number of iterations *MIter* used at the start of the optimization process to construct a front using NMM are 50 and 30 respectively, the performance of the algorithm is better. For the parameters of M-MMPA-GM, Table 4 shows the results obtained by different values for both *NSS* and the repetition times *Nt* on each solution. The results introduced in this table show the performance of the algorithm is convergent with the different values of those parameters with the exception of $X = 100$ and $Nt = 1$. Concerning the other algorithms, the values suggested in the

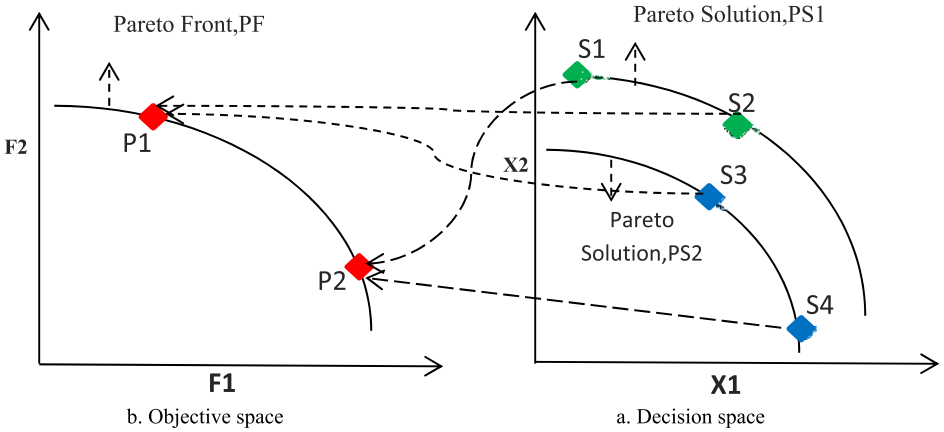


FIGURE 1. Illustration example of multi-modal multi-objective optimization problem.

TABLE 1. Description of test functions.

Descriptions of Cec2009, GLT					Descriptions of Cec2020				
F	Test function	No. of objectives	No. of variables	Reference	F	Test function	No. of objectives	No. of variables	Reference
F1	GLT1	2	10	[27]	F17	MMF1	2	2	[26]
F2	GLT2	2	10	[27]	F18	MMF1_e	2	2	[26]
F3	GLT3	2	10	[27]	F19	MMF2	2	2	[26]
F4	GLT4	2	10	[27]	F20	MMF4	2	2	[26]
F5	GLT5	3	10	[27]	F21	MMF5	2	2	[26]
F6	GLT6	3	10	[27]	F22	MMF7	2	2	[26]
F7	UF1	2	30	[25]	F23	MMF8	2	2	[26]
F8	UF2	2	30	[25]	F24	MMF10	2	2	[26]
F9	UF3	2	30	[25]	F25	MMF10_l	2	2	[26]
F10	UF4	2	30	[25]	F26	MMF11	2	2	[26]
F11	UF5	2	30	[25]	F27	MMF11_l	2	2	[26]
F12	UF6	2	30	[25]	F28	MMF12	2	2	[26]
F13	UF7	2	30	[25]	F29	MMF16	2	2	[26]
F14	UF8	3	30	[25]	F30	MMF16_l1	2	3	[26]
F15	UF9	3	30	[25]	F31	MMF16_l2	2	3	[26]
F16	UF10	3	30	[25]	F32	MMF16_l3	3	3	[26]

TABLE 2. IGD values obtained by the M-MMPA using different LC, and UC values from F1 to F6 (GLT).

LC=5, UC=20										
Algorithm	F	Best	Avg	worst	Std	F	Best	Avg	Worst	Std
M-MMPA	**F1**	0.0001687	0.0017480	0.0050457	0.0016484	**F4**	**0.0002307**	0.0030485	0.0087630	0.0035191
M-MMPA	**F2**	0.0012414	0.0014311	0.0024217	0.0002960	**F5**	0.0017954	0.0020118	0.0021916	**0.0001245**
M-MMPA	**F3**	0.0002633	0.0009537	0.0029627	0.0007530	**F6**	**0.0013462**	0.0014508	0.0015628	**0.0000549**
LC=10, UC=30										
Algorithm	F	Best	Avg	worst	Std	F	Best	Avg	Worst	Std
M-MMPA	**F1**	0.0001567	0.0015142	0.0054976	0.0016303	**F4**	0.0007020	0.0046566	0.0089270	0.0031399
M-MMPA	**F2**	0.0012868	0.0018987	0.0081546	0.0015040	**F5**	0.0018021	0.0019468	0.0022556	0.0001008
M-MMPA	**F3**	0.0002581	0.0007650	0.0014887	0.0003407	**F6**	0.0013811	0.0015083	0.0016650	0.0000732
LC=0, UC=50										
Algorithm	F	Best	Avg	worst	Std	F	Best	Avg	Worst	Std
M-MMPA	**F1**	**0.0001488**	**0.0005494**	**0.0033889**	**0.0007933**	**F4**	0.0002341	**0.0005115**	**0.0033990**	**0.0006867**
M-MMPA	**F2**	**0.0012420**	**0.0013121**	**0.0014353**	**0.0000424**	**F5**	**0.0016842**	**0.0019367**	**0.0022370**	0.0001370
M-MMPA	**F3**	**0.0001825**	**0.0002454**	**0.0006016**	**0.0000976**	**F6**	0.0013761	0.0015054	**0.0016375**	0.0000762

original papers have been used in our comparison except for the population size and the number of evaluations that are respectively set to 100 and 50000 for all algorithms to ensure a fair comparison. The final parameter values used in our comparison are introduced in Table 5. Note that, the maximum number of evaluations used for the two versions integrated by GM or NMM involves the number of evaluations used by those two strategies.

C. PERFORMANCE METRICS

The performance of each algorithm is evaluated based on two metrics: the inverted generational distance (IGD) is used to evaluate the convergence, and the generalized spread metric (GSM) is used to verify the coverage.

Inverted generational distance (IGD) metric: this metric [81] evaluates the convergence of the results by calculating the distance between the obtained Pareto front solutions and

TABLE 3. IGD values obtained by the M-MMPA-NMM using different *NSS*, and MIter values from F1 to F6 (GLT).

***NSS* =50, *MIter* =10**										
Algorithm	F	Best	Avg	worst	Std	F	Best	Avg	Worst	Std
M-MMPA-NMM	**F1**	0.0001252	0.0003804	0.0036008	0.0007445	**F4**	0.0002259	0.0004965	0.0017238	0.0004431
M-MMPA-NMM	**F2**	0.0012816	0.0014091	0.0029143	0.0003501	**F5**	0.0017744	0.0019411	0.0022707	**0.0001288**
M-MMPA-NMM	**F3**	0.0001824	0.0002880	0.0013807	0.0002558	**F6**	0.0013465	0.0015538	0.0019169	0.0001754
***NSS* =50, *MIter* =30**										
Algorithm	F	Best	Avg	worst	Std	F	Best	Avg	Worst	Std
M-MMPA-NMM	**F1**	**0.0001243**	0.0002137	**0.0005682**	**0.0001174**	**F4**	**0.0002260**	**0.0003636**	**0.0010063**	**0.0002383**
M-MMPA-NMM	**F2**	0.0012533	**0.0013016**	**0.0014371**	**0.0000443**	**F5**	**0.0016801**	**0.0019253**	**0.0021816**	0.0001339
M-MMPA-NMM	**F3**	0.0001820	0.0002566	0.0010158	**0.0001802**	**F6**	0.0012623	0.0015077	0.0017309	**0.0000936**
***NSS* =50, *UC*=40**										
Algorithm	F	Best	Avg	worst	Std	F	Best	Avg	Worst	Std
M-MMPA-NMM	**F1**	0.0001343	0.0002622	0.0010184	0.0002080	**F4**	0.0002347	0.0005857	0.0037761	0.0007924
M-MMPA-NMM	**F2**	0.0012606	0.0014824	0.0051439	0.0008404	**F5**	0.0017250	0.0019340	0.0020897	0.0001016
M-MMPA-NMM	**F3**	0.0001821	0.0002160	0.0004582	0.0000587	**F6**	0.0013554	0.0014880	0.0018023	0.0001047
***NSS* =20, *MIter* =30**										
Algorithm	F	Best	Avg	worst	Std	F	Best	Avg	Worst	Std
M-MMPA-NMM	**F1**	0.0001396	**0.0001996**	0.0004132	0.0000762	F4	0.0002263	0.0011454	0.0061121	0.0016275
M-MMPA-NMM	**F2**	0.0012632	0.0013516	0.0014775	0.0000644	F5	0.0017460	0.0019498	0.0024083	0.0001639
M-MMPA-NMM	**F3**	0.0001824	0.0002397	0.0004059	0.0000650	F6	**0.0011932**	**0.0014835**	**0.0017121**	0.0001005
***NSS* =30, *MIter* =30**										
Algorithm	F	Best	Avg	worst	Std	F	Best	Avg	Worst	Std
M-MMPA-NMM	**F1**	0.0001427	0.0003585	0.0037867	0.0007870	F4	0.0002264	0.0004611	0.0018599	0.0004491
M-MMPA-NMM	**F2**	0.0012602	0.0013396	0.0015414	0.0000810	F5	0.0016799	0.0019606	0.0021720	0.0001365
M-MMPA-NMM	**F3**	**0.0001816**	**0.0002304**	**0.0004012**	**0.0000647**	F6	0.0012901	0.0015010	0.0017401	0.0000999
***NSS* =60, *MIter* =30**										
Algorithm	F	Best	Avg	worst	Std	F	Best	Avg	Worst	Std
M-MMPA-NMM	**F1**	0.0001390	0.0003942	0.0038780	0.0008097	F4	0.0002231	0.0005269	0.0028147	0.0006596
M-MMPA-NMM	**F2**	**0.0012473**	0.0013800	0.0027239	0.0003099	F5	0.0017091	0.0019297	0.0020462	0.0000935
M-MMPA-NMM	**F3**	0.0001820	0.0002205	0.0003895	0.0000522	F6	0.0013683	0.0015271	0.0017216	0.0000773

TABLE 4. IGD values obtained by the M-MMPA-GM using different *NSS*, and Nt values from F1 to F6 (GLT).

***NSS* =50, *Nt* =2, number of Evaluations=100**										
Algorithm	F	Best	Avg	worst	Std	F	Best	Avg	Worst	Std
M-MMPA-GM	**F1**	0.0000745	0.0001112	0.0005717	0.0001084	**F4**	0.0002013	0.0005258	0.0026804	0.0016299
M-MMPA-GM	**F2**	0.0011601	0.0012370	0.0013109	0.0000378	**F5**	0.0016134	**0.0017841**	**0.0019665**	**0.0000932**
M-MMPA-GM	**F3**	0.0001775	0.0002142	0.0004322	0.0000738	**F6**	0.0012667	**0.0013829**	0.0016205	**0.0000995**
***NSS* =20, *Nt* =5, number of Evaluations=100**										
Algorithm	F	Best	Avg	worst	Std	F	Best	Avg	Worst	Std
M-MMPA-GM	**F1**	0.0000686	0.0000769	0.0002102	0.0000306	**F4**	**0.0001985**	**0.0002009**	**0.0002036**	**0.0000015**
M-MMPA-GM	**F2**	0.0011792	0.0012472	0.0013001	0.0000316	**F5**	0.0015214	0.0017920	0.0020194	0.0001142
M-MMPA-GM	**F3**	**0.0001775**	**0.0001835**	**0.0002203**	**0.0000111**	**F6**	0.0012850	0.0013980	0.0015151	0.0000641
***NSS* =25, *Nt* =4, number of Evaluations=100**										
Algorithm	F	Best	Avg	worst	Std	F	Best	Avg	Worst	Std
M-MMPA-GM	**F1**	**0.0000697**	**0.0000715**	**0.0000744**	**0.0000014**	**F4**	0.0001991	0.0008419	0.0067978	0.0018213
M-MMPA-GM	**F2**	**0.0011937**	**0.0012317**	**0.0012791**	**0.0000215**	**F5**	0.0016355	0.0018370	0.0021859	0.0001374
M-MMPA-GM	**F3**	0.0001775	0.0002582	0.0007003	0.0001693	**F6**	**0.0011166**	0.0013831	0.0015469	0.0001102
***NSS* =100, *Nt* =1, number of Evaluations=100**										
Algorithm	F	Best	Avg	worst	Std	F	Best	Avg	Worst	Std
M-MMPA-GM	**F1**	0.0000851	0.0001024	0.0001432	0.0000151	**F4**	0.0002077	0.0013265	0.0084609	0.0025238
M-MMPA-GM	**F2**	0.0011958	0.0012523	0.0013026	0.0000296	**F5**	0.0016461	0.0018467	0.0020406	0.0001307
M-MMPA-GM	**F3**	0.0001784	0.0002412	0.0005569	0.0001029	**F6**	0.0013164	0.0014145	0.0016648	0.0000900

the true Pareto solutions, and is formulated as follows:

$$IGD = \frac{\sum_{i=1}^{N} d_i}{N} \tag{24}$$

$$d_i = \sqrt{\sum_{j=1}^{k} (f_{i,j}^{x} - f_j^{true})^2} \tag{25}$$

where N is the number of non-dominated solutions obtained. d_i is the smallest Euclidean distance between the i^{th} solutions and the true Pareto optimal solutions. k specifies the number of objectives in the test function. $f_{i,j}^{x}$ is the j^{th} the objective function of the i^{th} The Pareto solution obtained by algorithm x. f_j^{true} is the nearest Pareto solution in the true Pareto front from $f_{i,j}^{x}$. When IGD is equal to 0, all the obtained non-dominated solutions are in the true Pareto front.

Generalized spread metric (GSM): The coverage of the obtained Pareto optimal solutions is measured and qualified via this metric [39] as:

$$GSM = \frac{\sum_{j=1}^{k} d_j^{ex} + \sum_{p=2}^{N} \left| d_p - \bar{\bar{d}} \right|}{\sum_{j=1}^{k} d_j^{ex} + (k-1)\bar{\bar{d}}} \tag{26}$$

where d_j^{ex} is the Euclidian distance between the true Pareto front and the obtained Pareto front, d_p is the Euclidian

TABLE 5. Parameter setting for the algorithms.

Parameters using for NSGA-II [75]		Parameters using for SMPSO [77]	
Number of runs	20	Number of runs	20
Population size	10	Population size	100
The maximum number of evaluations	50000	The maximum number of evaluations	50000
Selection of Parents	binary tournament	Selection of Parents	binary tournament
Recombination	pc = 0.9	Recombination	pc = 0.9
Mutation	polynomial, pm = 1.0/n	Mutation	polynomial, pm = 1.0/n
Parameters using for MMPA		**Parameters using for NSGA-III** [76]	
Number of runs	20	Number of runs	20
Population size	100	Population size	100
The maximum number of evaluations	50000	The maximum number of evaluations	50000
Archive size	100	Selection of Parents	binary tournament
		Recombination	simulated binary, pc = 0.9
		Mutation	polynomial, pm = 1.0/n
Parameters using for M-MMPA			
Number of runs	20		
Population size	100		
The maximum number of evaluations	50000		
Archive size	100		
LC	0		
UC	40		
Parameters using for M-MMPA-GM		**Parameters using for GDE3 [78]**	
Number of runs	20	Number of runs	20
Population size	100	Population size	100
The maximum number of evaluations	50000	The maximum number of evaluations	50000
Nt	2	Recombination	Differential Evolution , pc = 0.9
Archive size	100	Archive size	100
NSS	50		
EP	0.15		
Parameters using for M-MMPA-NMM		**Parameters using for CDG [79]**	
Number of runs	20	Number of runs	20
Population size	100	Population size	100
The maximum number of evaluations	50000	The maximum number of evaluations	50000
$MIter$	30	Recombination	Differential Evolution , pc = 0.9
Archive size	100	Archive size	100
NSS	50		

distance between two consecutive points of the obtained non-dominated solutions, and $\bar{d}$ is the average of d_p.

In both metrics, the lower value is the best.

V. EXPERIMENTAL STUDIES

Statistical results play a significant role in evaluating the quality of the algorithms. As a result, we present the best, worst, average (Avg), standard deviation (Std), and rank (R) values as statistical measures to compare the results obtained by each algorithm. Additionally, scatterplots and boxplots have been used to illustrate comparative performances. In the following sections, the performance of the algorithms on GLT, CEC 2009, and CEC 2020 are discussed separately:

1) Experiment 1: Comparison and discussion for GLT.
2) Experiment 2: Comparison and discussion for CEC 2009.
3) Experiment 3: Comparison and discussion for CEC 2020.

A. EXPERIMENT 1: COMPARISON AND DISCUSSION FOR GLT

Table 6 shows the statistical results obtained by each algorithm on GLT problems for the IGD metric. It should be noted that M-MMPA-GM, M-MMPA, and M-MMPA-NMM have exceptional results for F1, F3, and F4 test functions. Concerning F2, F5, and F6 test functions, as shown in Table 6, M-MMPA-GM outperforms all algorithms for all the statistical measures (best, worst, avg, std) except for F5, in which the proposed algorithms were outperformed by NSGA-III. The statistical results indicate the promising performances of the proposed algorithms in solving GLT problems—in particular, M-MMPA-GM outperforms all other algorithms in all GLT test functions except F5. This superiority stems from the ability of the Gaussian-based mutation strategy in achieving high convergence towards the true Pareto optimal front.

The average of IGD values obtained by all the compared algorithms within 20 independent runs is shown in Fig. 2,

TABLE 6. IGD values of the objective space obtained by the proposed algorithms from F1 to F6 (GLT).

Algorithm	F	Best	Avg	Worst	Std	R	F	Best	Avg	Worst	Std	R
MMPA	**F1**	0.0001250	0.0010578	0.0032050	0.0009257	4	**F4**	0.0002137	0.0023502	0.0087560	0.0023488	4
M-MMPA		0.0001243	0.0001829	0.0003493	0.0000597	2		0.0002252	0.0006522	0.0044208	**0.0010283**	2
M-MMPA-GM		**0.0000809**	**0.00009673**	**0.0001103**	**0.0000155**	**1**		**0.0002021**	**0.0005067**	**0.0022453**	0.0021751	**1**
M-MMPA-NMM		0.0001232	0.0002264	0.0005901	0.0001086	3		0.0002163	0.0013843	0.0085426	0.0024431	3
SMPSO**[77]**		0.0022860	0.0040986	0.0061087	0.0010052	5		0.0006587	0.0080702	0.0124557	0.0022164	6
GDE3**[78]**		0.0045491	0.0051119	0.0055188	0.0002423	6		0.0087433	0.0106102	0.0170807	0.0027456	8
NSGA-II **[75]**		0.0019791	0.0051223	0.0060427	0.0009612	7		0.0021541	0.0100511	0.0172625	0.0035839	7
NSGA-III**[76]**		0.0047443	0.0060756	0.0064065	0.0005573	9		0.0088245	0.0140489	0.0181179	0.0036527	9
CGD**[79]**		0.0000999	0.0005662	0.0017875	0.0005049	8		0.0008504	0.0054067	0.0082611	0.0017771	5
MMPA	**F2**	0.0012466	0.0012839	0.0013565	0.0000264	2	**F5**	0.0017768	0.0020398	0.0023382	0.0001605	5
M-MMPA		0.0012358	0.0013173	0.0014435	0.0000467	4		0.0016279	0.0019113	0.0021303	0.0001303	2
M-MMPA-GM		**0.0011892**	**0.0012430**	**0.0013022**	**0.0000304**	**1**		0.0017022	**0.0018757**	**0.0019940**	**0.0000938**	**1**
M-MMPA-NMM		0.0012497	0.0013053	0.0014750	0.0000486	3		0.0017827	0.0019579	0.0023195	0.0001315	3
SMPSO**[77]**		0.0016327	0.0019688	0.0027669	0.0002819	5		0.0020887	0.0025430	0.0029695	0.0002028	8
GDE3**[78]**		0.0034301	0.0063673	0.0139419	0.0034808	7		0.0019156	0.0022147	0.0024662	0.0001337	6
NSGA-II **[75]**		0.0055442	0.0205648	0.0409157	0.0120517	8		0.0019155	0.0023556	0.0047575	0.0006548	7
NSGA-III**[76]**		0.0230777	0.0528013	0.0911649	0.0237281	9		**0.0014151**	0.0019596	0.0033371	0.0004175	4
CGD**[79]**		0.0031441	0.0053746	0.0103642	0.0018572	6		0.0020380	0.0081948	0.0124031	0.0033510	9
MMPA	**F3**	0.0001818	0.0002647	0.0010118	0.0001807	4	**F6**	0.0012812	0.0014900	0.0016440	0.0000949	4
M-MMPA		0.0001853	0.0002446	0.0003900	0.0000664	3		0.0013489	0.0015204	0.0017451	0.0000845	5
M-MMPA-GM		**0.0001789**	**0.0002004**	**0.0004212**	**0.0000547**	**1**		**0.0012174**	**0.0013749**	**0.0015421**	**0.0000932**	**1**
M-MMPA-NMM		0.0001821	0.0002354	0.0008243	0.0001396	2		0.0012770	0.0014386	0.0017843	0.0001223	2
SMPSO**[77]**		0.0007093	0.0022947	0.0051639	0.0012035	5		0.0016077	0.0018100	0.0020402	0.0001217	8
GDE3**[78]**		0.0016230	0.0045828	0.0059815	0.0013651	8		0.0014467	0.0016630	0.0018502	0.0000995	7
NSGA-II **[75]**		0.0020109	0.0038793	0.0061140	0.0013499	7		0.0014263	0.0015707	0.0017485	0.0000867	6
NSGA-III**[76]**		0.0036168	0.0063446	0.0071033	0.0007224	9		0.0013682	0.0014481	0.0015164	0.0000403	3
CGD**[79]**		0.0016113	0.0030195	0.0049349	0.0009541	6		0.0011485	0.0038129	0.0124612	0.0035114	9

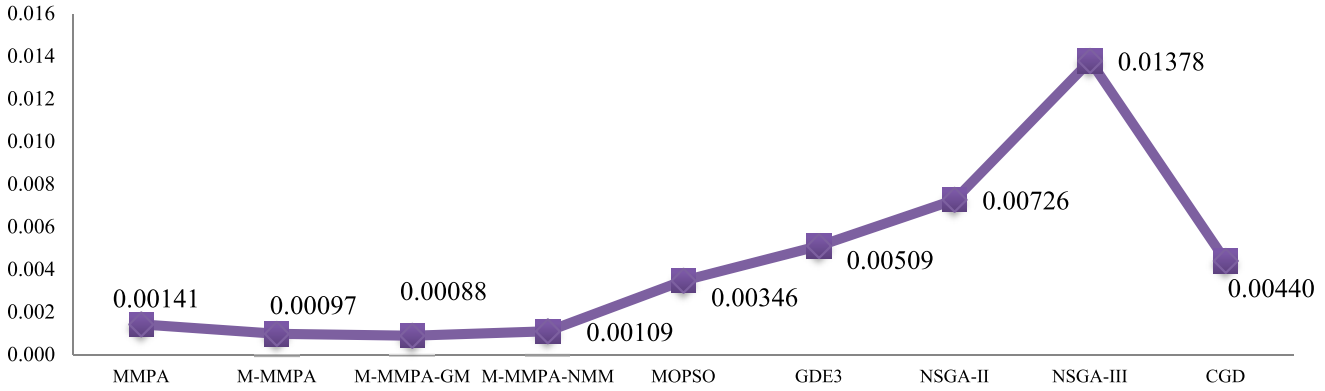


FIGURE 2. The average of IGD values on GLT problems (F1-F6).

from which it can be inferred that M-MMPA-GM performs the best with a value of 0.00088, while NSGA-III is worst with a value of 0.01378, with the worst performance on GLT problems.

Table 7 shows that the proposed algorithms achieve better coverage in all test functions, with the best coverage achieved by M-MMPA-GM. The average GSM values obtained on GLT problems within 20 independent runs are shown in Fig. 3, from which M-MMPA-GM can be seen to be the best with a value of 0.2768 and NSGA-III is the worst with a value of 0.9750.

Fig. 4a and 4b show a comprehensive overview of the Pareto optimal front obtained by each algorithm on GLT test functions. Fig. 4a and 4b show the obtained Pareto front results of the proposed algorithms and the other algorithms in which M-MMPA-GM can be seen to achieve better convergence and coverage than the other algorithms. In addition, the solutions obtained by M-MMPA-GM are distributed extensively on the whole Pareto front.

B. EXERIEMNT 2: COMPARISON AND DISCUSSION FOR CEC 2009

Table 8 demonstrates the statistical results obtained by each algorithm on CEC 2009 problems for the IGD metric, in which M-MMPA-GM achieves exceptional results for F7, F8, and F13 test functions. With respect to the other test functions, as shown in Table 8, M-MMPA-GM outperforms all algorithms for F9, F10, F11, and F12 test functions with converged results with some other algorithms, and M-MMPA-NMM is best for F14, and F16, while GDE3 achieves the best value for F15. The statistical results

TABLE 7. GSM values obtained by the proposed algorithms from F1 to F6 (GLT).

Algorithm	F	Best	Avg	worst	Std	R	F	Best	Avg	Worst	Std	R
MMPA	**F1**	0.1563696	0.0941507	0.0668343	0.0269933	2	**F2**	**0.2229825**	0.1613296	0.1225103	0.0240167	2
M-MMPA		0.1734243	0.1163223	0.0749059	0.0250151	4		0.2940404	0.1933311	0.1196703	0.0407047	3
M-MMPA-GM		**0.1561744**	**0.0901954**	**0.0650252**	**0.0180504**	**1**		0.2414409	**0.1530450**	**0.1134459**	**0.0279229**	**1**
M-MMPA-NMM		0.2522654	0.1138474	0.0665631	0.0474023	3		0.3628802	0.2016486	0.1387248	0.0493286	4
SMPSO[77]		1.7872458	0.8840440	0.3065340	0.5025735	8		1.1782443	0.6199224	0.2283829	0.3355071	5
GDE3[78]		1.3761342	0.3989113	0.2611904	0.2304665	5		1.6541125	0.7378641	0.2771873	0.4941891	7
NSGA-II [75]		0.6699407	0.4022522	0.3042369	0.0819958	6		1.8395223	0.6464806	0.3408272	0.3788576	6
NSGA-III[76]		1.6987663	1.0456848	0.7954055	0.2075773	9		1.4282600	1.0914607	0.6422287	0.1949061	9
CGD[79]		0.8321289	0.6154108	0.4425220	0.0891711	7		1.0014913	0.7997474	0.4165222	0.1379899	8
MMPA	**F3**	0.5103035	0.4738569	0.4224673	0.0244993	3	**F4**	0.5204173	0.4350763	0.3793097	0.0331909	4
M-MMPA		0.5327013	0.4799420	0.4255208	0.0267503	4		0.4843983	0.4256285	0.3602556	0.0347421	3
M-MMPA-GM		**0.4666100**	**0.4226313**	**0.3845977**	**0.0240845**	**1**		0.5014056	0.4421960	0.3752596	0.0377314	7
M-MMPA-NMM		0.5121267	0.4719795	0.3955924	0.0297873	2		**0.4766118**	**0.4023950**	0.3596896	**0.0310355**	**1**
SMPSO[77]		0.6517102	0.5555851	0.4776364	0.0503897	5		0.5293885	0.4330054	0.3262553	0.0484880	5
GDE3[78]		1.0790770	0.8673786	0.6460076	0.1327612	8		0.4937670	0.4367535	0.3818338	0.0350471	6
NSGA-II [75]		0.9413718	0.6721575	0.4710920	0.1084418	6		0.6346780	0.5320337	0.4556137	0.0528337	8
NSGA-III[76]		1.6131838	1.2358713	0.8832602	0.2109357	9		0.7058911	0.4061732	**0.3123060**	0.0841808	2
CGD[79]		0.8580392	0.7716182	0.6586329	0.0638670	7		0.8796877	0.6680494	0.4451701	0.1024848	9
MMPA	**F5**	0.1516059	0.1086280	0.0759000	0.0162681	2	**F6**	**0.5221106**	0.4412660	0.3742711	0.0396732	4
M-MMPA		0.1930730	0.1267456	0.0942631	0.0244349	3		0.5422116	**0.4284306**	**0.3143432**	**0.0642778**	**1**
M-MMPA-GM		**0.1481641**	**0.1135630**	**0.0844015**	**0.0180198**	**1**		0.5417463	0.4392709	0.3504952	0.0458329	2
M-MMPA-NMM		0.1769825	0.1335364	0.1021486	0.0219807	4		0.5241254	0.4406990	0.3500763	0.0544634	3
SMPSO[77]		0.6174023	0.3395992	0.1270337	0.1405283	5		0.6268565	0.5570859	0.4804203	0.0411604	5
GDE3[78]		0.7340664	0.3966018	0.2444910	0.1447455	6		0.7601554	0.6668918	0.5495410	0.0529210	8
NSGA-II [75]		1.0515297	0.4521445	0.3450668	0.1425109	7		0.7448639	0.6265161	0.5163707	0.0546127	6
NSGA-III[76]		1.6067758	1.3537920	1.1854699	0.1195756	9		0.8963707	0.7172443	0.5145951	0.1033697	9
CGD[79]		1.0777959	0.9527988	0.8148617	0.0622743	8		0.9682045	0.6313286	0.4081635	0.1379216	7

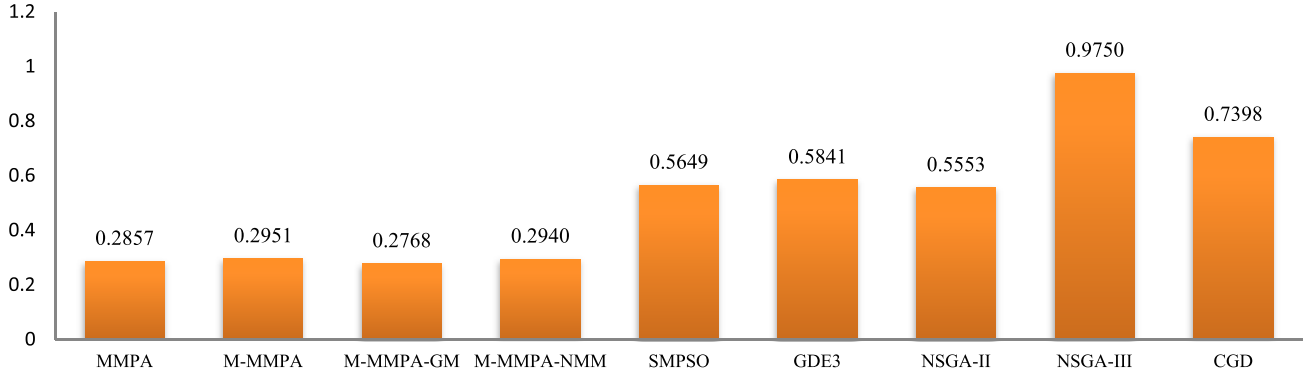


FIGURE 3. The average of GSM values on GLT problems (F1-F6).

indicate promising performances of the proposed algorithms in solving CEC 2009 problems, especially M-MMPA-GM which achieves dramatic results on several test functions—again, those dramatic results are achieved as a result of using the Gaussian-based mutation strategy to achieve better convergence towards the true Pareto optimal front.

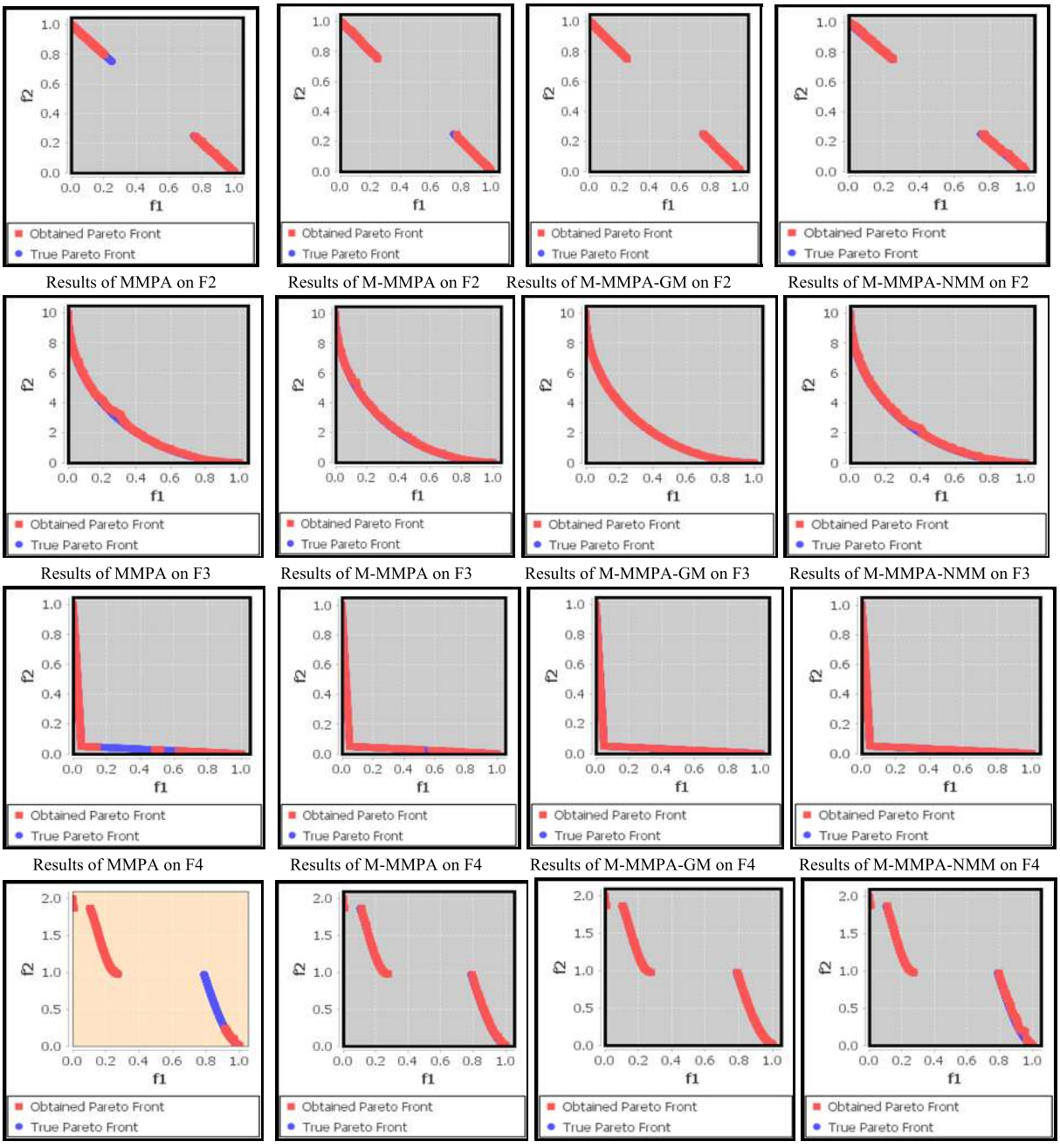
The average of IGD values obtained by all algorithms within 20 independent runs is shown in Fig. 5 from which it can be concluded that M-MMPA-GM is the best with a value of 0.01232, while SMPSO is worst with a value of 0.04387.

From Table 9 it can be seen that M-MMPA-GM achieves better coverage in 4 out of 10 test functions, while MMPA and M-MMPA-NMM also achieve better coverage 4 out of 10 test functions, and CGD in F15 and F16. The average of G*SM* obtained by the algorithms on GLT problems within 20 independent runs is shown in Fig. 6 which shows that M-MMPA-NMM is the best with a value of 0.59218 and NSGA-III is worst with a value of 0.80728.

Fig. 7a and 7b show a comprehensive overview of the Pareto optimal front obtained by each algorithm on CEC 2009 test functions. The figures show that the proposed algorithms achieve better convergence and coverage towards the true Pareto optimal, especially M-MMPA-GM that produces a more extensive Pareto front although there is a gap in the front for some functions.

C. EXPERIMENT 3: COMPARISON AND DISCUSSION FOR CEC 2020

Table 10 elaborates the statistical results obtained by each algorithm on CEC 2020 problems for the IGD metric, in which M-MMPA-GM has better results for F20 and MMPA has better results for F17, F19, F21, F22, F23, F24, and F28. M-MMPA outperforms all algorithms in F18 and F26 and M-MMPA-NMM outperforms all the other algorithms in F25 and F27. Unfortunately, our proposed algorithms does not outperform the CGD algorithm for functions

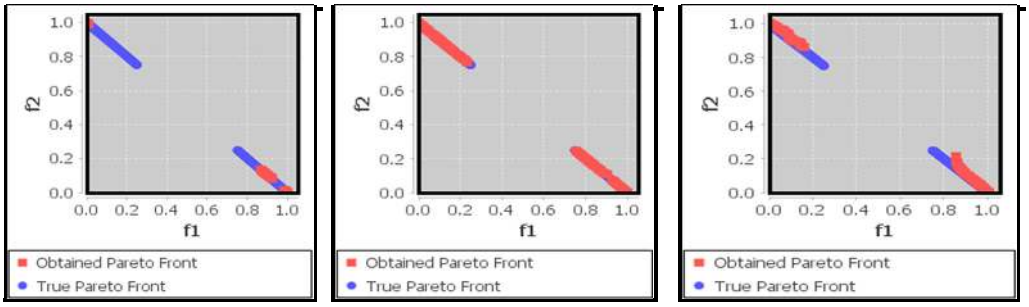


Results of MMPA on F2 | Results of M-MMPA on F2 | Results of M-MMPA-GM on F2 | Results of M-MMPA-NMM on F2

Results of MMPA on F3 | Results of M-MMPA on F3 | Results of M-MMPA-GM on F3 | Results of M-MMPA-NMM on F3

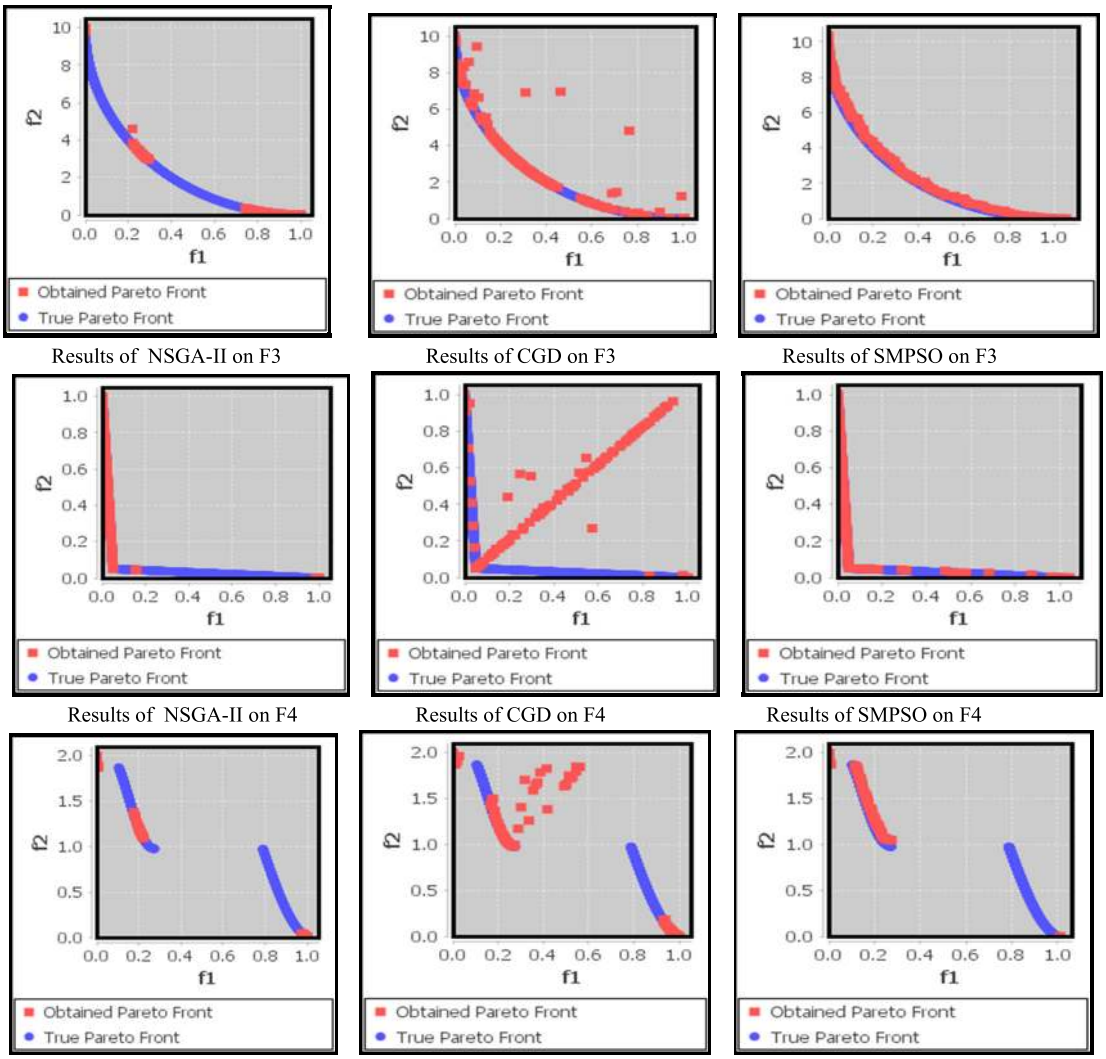
Results of MMPA on F4 | Results of M-MMPA on F4 | Results of M-MMPA-GM on F4 | Results of M-MMPA-NMM on F4

a. Pareto optimal front obtained by proposed algorithms.



Results of NSGA-II on F2 | Results of CGD on F2 | Results of SMPSO on F2

FIGURE 4. Pareto optimal front obtained by all algorithms.



b. Pareto optimal front obtained by CGD, NSGA-II, and SMPSO algorithms

FIGURE 4. ***(Continued.)*** **Pareto optimal front obtained by all algorithms.**

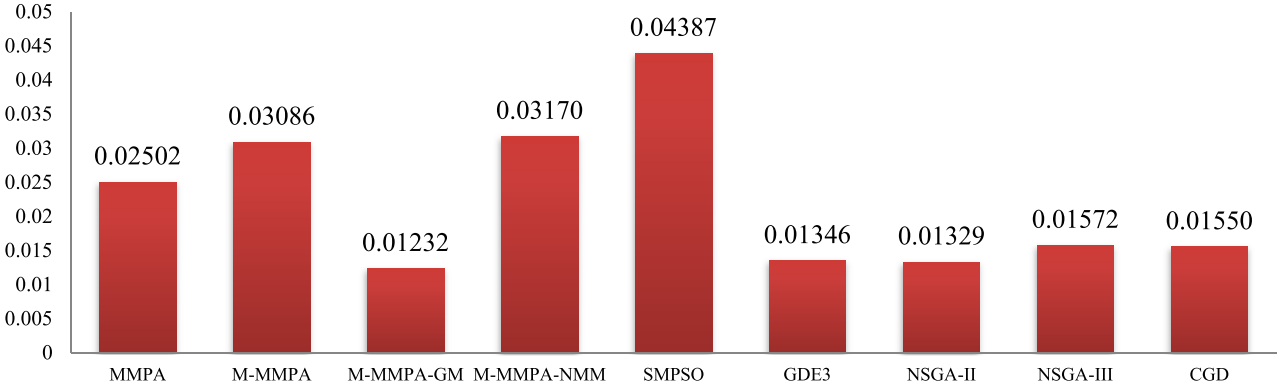


FIGURE 5. **The average of the IGD values on CEC 2009 problems (F7-F16).**

from F29 to F32. Generally, MMPA performs best for CEC 2020 for 7 out of 16 test functions. The statistical results indicate the promising performances of the proposed algorithms in solving CEC 2020 problems since the proposed versions could outperform all other algorithms in 12 out of 16 test functions.

TABLE 8. IGD values obtained on problems from F7 to F16.

Algorithm	F	Best	Avg	Worst	Std	R	F	Best	Avg	Worst	Std	R
MMPA	F7	0.0018102	0.0027486	0.0037660	0.0005515	4	F12	0.0095648	0.0132409	0.0175354	0.0022588	5
M-MMPA		0.0014182	0.0021347	0.0028237	0.0003579	3		0.0075587	0.0151614	0.0230417	0.0032669	7
M-MMPA-GM		**0.0008061**	**0.0013283**	**0.0017340**	**0.0002521**	1		0.0056450	**0.0080850**	**0.0130928**	**0.0018991**	1
M-MMPA-NMM		0.0017350	0.0020258	0.0025192	0.0001977	2		0.0135470	0.0172767	0.0214780	0.0024387	8
SMPSO[77]		0.0026303	0.0043124	0.0057396	0.0008625	9		0.0157692	0.0202984	0.0294867	0.0036749	9
GDE3[78]		0.0031112	0.0034248	0.0035909	0.0001504	6		**0.0036691**	0.0082611	0.0131267	0.0034086	2
NSGA-II [75]		0.0018487	0.0029094	0.0038128	0.0004820	5		0.0040766	0.0089163	0.0162157	0.0026081	3
NSGA-III[76]		0.0030626	0.0035362	0.0042634	0.0003205	7		0.0097926	0.0138095	0.0225066	0.0029353	6
CGD[79]		0.0018453	0.0038013	0.0090470	0.0016178	8		0.0033683	0.0106020	0.0307325	0.0090285	4
MMPA	F8	0.0010093	0.0013238	0.0015726	0.0001340	4	F13	0.0016851	0.0051901	0.0098723	0.0029520	6
M-MMPA		0.0009692	0.0011845	0.0014220	0.0001258	2		0.0015393	0.0023832	0.0054061	0.0008965	5
M-MMPA-GM		**0.0006571**	**0.0007893**	**0.0013258**	**0.0001338**	1		**0.0005614**	**0.0007879**	**0.0011891**	**0.0001692**	1
M-MMPA-NMM		0.0009128	0.0011979	0.0014981	0.0001374	3		0.0013214	0.0018814	0.0037900	0.0005561	2
SMPSO[77]		0.0017032	0.0020050	0.0024951	0.0002134	8		0.0015746	0.0021486	0.0033454	0.0003704	3
GDE3[78]		0.0012784	0.0017934	0.0023167	0.0003016	6		0.0019367	0.0021887	0.0025525	0.0001396	4
NSGA-II [75]		0.0011883	0.0014335	0.0017550	0.0001624	5		0.0010594	0.0027984	0.0154670	0.0036576	6
NSGA-III[76]		0.0028658	0.0033244	0.0038084	0.0002432	9		0.0027742	0.0087767	0.0215468	0.0056714	8
CGD[79]		0.0011494	0.0019389	0.0055645	0.0012424	7		0.0012537	0.0058604	0.0258187	0.0079038	7
MMPA	F9	0.0033956	0.0049030	0.0063090	0.0009086	2	F14	0.0030797	0.0033916	0.0036552	0.0001663	4
M-MMPA		0.0047616	0.0072404	0.0109252	0.0014366	4		0.0029641	0.0034026	0.0049281	0.0004232	5
M-MMPA-GM		**0.0038775**	**0.0049675**	**0.0057751**	**0.0004695**	1		0.0034354	0.0041397	0.0050564	0.0004536	8
M-MMPA-NMM		0.0050636	0.0069408	0.0090263	0.0007889	3		0.0026941	**0.0027174**	**0.0027564**	**0.0000174**	1
SMPSO[77]		0.0068185	0.0077123	0.0083589	0.0004121	5		0.0027470	0.0028950	0.0030643	0.0000922	2
GDE3[78]		0.0098673	0.0113776	0.0119023	0.0004827	8		0.0033430	0.0036174	0.0039509	0.0001585	7
NSGA-II [75]		0.0082194	0.0091081	0.0107639	0.0006664	6		0.0032961	0.0058248	0.0090254	0.0013512	9
NSGA-III[76]		0.0109261	0.0132050	0.0153768	0.0012880	9		0.0023223	0.0032653	0.0048561	0.0006421	3
CGD[79]		0.0046486	0.0093366	0.0131374	0.0026269	7		**0.0020843**	0.0035286	0.0052094	0.0006173	6
MMPA	F10	0.0020559	0.0022424	0.0024350	0.0001037	5	F15	0.0029114	0.0039048	0.0049961	0.0004724	4
M-MMPA		0.0020126	0.0022913	0.0027611	0.0001684	8		0.0034969	0.0041660	0.0049090	0.0004069	6
M-MMPA-GM		**0.0014127**	**0.0015016**	**0.0016141**	**0.0000496**	1		0.0038185	0.0046239	0.0056951	0.0004668	8
M-MMPA-NMM		0.0019615	0.0022874	0.0025440	0.0001454	7		0.0032915	0.0040585	0.0050393	0.0004588	5
SMPSO[77]		0.0015508	0.0017783	0.0019360	0.0000987	5		0.0032965	0.0044296	0.0051607	0.0004711	7
GDE3[78]		0.0014513	0.0015193	0.0016535	0.0000487	2		**0.0015259**	**0.0021913**	**0.0042121**	**0.0006157**	1
NSGA-II [75]		0.0014791	0.0015270	0.0017358	0.0000547	3		0.0045759	0.0079017	0.0111830	0.0015293	9
NSGA-III[76]		0.0026927	0.0029645	0.0032464	0.0001427	9		0.0021354	0.0031903	0.0043628	0.0005288	2
CGD[79]		0.0014928	0.0016742	0.0019754	0.0001195	4		0.0021878	0.0034574	0.0044781	0.0006258	3
MMPA	F11	0.1356256	0.1935962	0.2981067	0.0401158	6	F16	0.0117740	0.0196741	0.0255276	0.0038055	5
M-MMPA		0.1352011	0.2461359	0.3552305	0.0604431	7		0.0169159	0.0245347	0.0375782	0.0052792	7
M-MMPA-GM		0.0501352	**0.0623422**	0.0963804	**0.0107829**	1		0.0250302	0.0346333	0.0466721	0.0054453	9
M-MMPA-NMM		0.2025627	0.2755425	0.3713376	0.0454115	8		**0.0028168**	**0.0030347**	**0.0034228**	**0.0001625**	1
SMPSO[77]		0.2071166	0.3897910	0.6156810	0.1297201	9		0.0028169	0.0033502	0.0044765	0.0004439	2
GDE3[78]		0.0451995	0.0840944	0.1075560	0.0133907	3		0.0128500	0.0161182	0.0213068	0.0018728	4
NSGA-II [75]		**0.0449126**	0.0651658	**0.0778560**	0.0108565	2		0.0106789	0.0273012	0.0453101	0.0073404	8
NSGA-III[76]		0.0548775	0.0986084	0.2114664	0.0358399	5		0.0039408	0.0064971	0.0130536	0.0022762	3
CGD[79]		0.0418390	0.0933711	0.2060299	0.0412263	4		0.0114269	0.0214387	0.0276572	0.0038398	6

The average of IGD values obtained by all the compared algorithms within 20 independent runs is shown in Fig. 8 from which it can be concluded that for CEC2020 M-MMPA-NMM is the best with a value of 0.00377, while SMPSO is worst with a value of 0.00547.

Table 11 shows that MMPA achieves better coverage in 6 out of 16 test functions, while M-MMPA and M-MMPA-GM outperform all others in 3 out of 16 test functions. Unfortunately, SMPSO achieves better coverage in 3 test functions, and CGD in F29, F30, F31and F32. The average of G*SM* obtained by the algorithms on CEC 2020 problems within 20 independent runs is shown in Fig. 9. As seen in Fig. 9, MMPA is best with a value of 0.34048, and NSGA-III is worst with a value of 0.80894.

Fig. 10a and 10b show the true Pareto front and the obtained Pareto optimal front obtained by each algorithm on CEC 2020 test functions. Fig. 10 shows the obtained fronts using CGD are generally less distributed and have less convergence than the other algorithms. Meanwhile, NSGA-II can be seen to achieve better coverage but not extensively on the whole front of the test functions while M-MMPA-GM, M-MMPA, MMPA, and SMPSO cover the whole Pareto front extensively.

D. COMPUTATIONAL COST AND OTHER COMPARISONS

In this section, the proposed algorithms will be extensively compared with three additional algorithms: modified Indicator based Evolutionary Algorithm (MIBEA) [82], SMSEMOA: Multiobjective selection based on dominated hypervolume [83], and multiobjective PSO with decomposition (DMOPSO) [84], in term of IGD metric on GLT test problems to show clearly the effectiveness of

TABLE 9. GSM values obtained by the proposed algorithms from F7 to F16.

Algorithm	F	Worst	Avg	Best	Std	R	F	Worst	Avg	Best	Std	R
MMPA	**F7**	1.0473880	0.6304883	0.3812245	0.1905918	5	**F12**	0.9928209	0.8613206	0.7275965	0.0787213	3
M-MMPA		0.8119934	0.5159496	0.2544604	0.1274284	2		1.1188837	0.8529698	0.4671080	0.1306118	2
M-MMPA-GM		**0.7141012**	**0.4548332**	**0.2915371**	**0.1388554**	**1**		1.4667738	0.9220335	0.6457224	0.1874103	6
M-MMPA-NMM		1.0235703	0.5562589	0.3276116	0.1861681	4		**0.9592607**	**0.8126195**	**0.6156029**	**0.0913608**	**1**
SMPSO**[77]**		1.0335941	0.7584112	0.5713613	0.1244627	6		1.3603863	0.9577031	0.7827922	0.1373567	9
GDE3**[78]**		0.8749058	0.5232330	0.2776471	0.1651738	3		1.2930154	0.9057325	0.6574796	0.1625421	4
NSGA-II **[75]**		1.3553291	0.9569242	0.4851627	0.1917666	9		1.3802440	0.9379259	0.7554724	0.1501040	7
NSGA-III**[76]**		1.1083227	0.8987938	0.4756237	0.1470999	7		1.3638788	0.9387960	0.6645879	0.1434179	8
CGD**[79]**		1.1614078	0.9556179	0.8045883	0.0988527	8		1.0664398	0.9001683	0.7876346	0.0883357	5
MMPA	**F8**	0.3998494	0.2483175	0.1809330	0.0616636	3	**F13**	0.9593635	0.4800564	0.1927383	0.2450242	5
M-MMPA		0.3501116	0.2461869	0.1828292	0.0418304	2		0.5264519	0.3573439	0.2561505	0.0676231	4
M-MMPA-GM		**0.3268123**	**0.2364761**	**0.1603514**	**0.0415511**	**1**		**0.3185755**	**0.2402354**	**0.1705989**	**0.0360319**	**1**
M-MMPA-NMM		0.3325181	0.2517710	0.1937618	0.0380045	4		0.5551396	0.3509863	0.2632225	0.0771476	2
SMPSO**[77]**		0.7004521	0.3991025	0.2856377	0.0895100	5		1.1780082	0.6104365	0.4224758	0.1753472	7
GDE3**[78]**		0.6355184	0.4217631	0.3143376	0.0787051	6		0.4628190	0.3536540	0.2538210	0.0592253	3
NSGA-II **[75]**		0.6027366	0.4322696	0.2923573	0.0744404	7		1.1048685	0.5285496	0.3081096	0.2198200	6
NSGA-III**[76]**		0.9747574	0.7252055	0.5134637	0.1365761	8		1.2045872	0.8918288	0.5103668	0.1964982	9
CGD**[79]**		1.2151878	1.0315737	0.8113199	0.1118476	9		0.9987414	0.8485370	0.6385390	0.1069059	8
MMPA	**F9**	**0.7245413**	**0.5074838**	**0.3125576**	**0.1071885**	**1**	**F14**	0.8337568	0.6717016	0.5208327	0.0796930	7
M-MMPA		1.1676646	0.6988885	0.2430340	0.2211836	4		0.9379514	0.7010304	0.5470938	0.1098991	8
M-MMPA-GM		1.0918767	0.8415648	0.6106019	0.1167897	5		0.8104302	0.6647318	0.5441325	0.0700299	6
M-MMPA-NMM		1.1056788	0.6155071	0.3489776	0.1875268	3		0.6759039	0.6022734	0.5018356	0.0425114	5
SMPSO**[77]**		0.8952249	0.5267110	0.2888305	0.1651262	2		0.6083637	0.5258858	0.4477067	0.0443331	1
GDE3**[78]**		1.0825641	0.7747672	0.4598761	0.1778858	6		0.8203488	0.6558113	0.5585346	0.0740150	2
NSGA-II **[75]**		0.9433996	0.8344467	0.6841951	0.0651998	7		0.6115742	0.5308280	0.4481646	0.0468515	3
NSGA-III**[76]**		1.1397897	0.8828564	0.6783392	0.0924986	8		1.0327549	0.7683069	0.5108117	0.1272099	9
CGD**[79]**		1.2633963	1.0141856	0.9365071	0.0781077	9		0.7134972	0.5446310	0.3502557	0.0879886	4
MMPA	**F10**	0.4215462	0.3156361	0.2342948	0.0512885	5	**F15**	0.8241401	0.6583714	0.4987377	0.0891951	5
M-MMPA		0.3928214	0.2780903	0.2004234	0.0449310	3		0.9221415	0.6597969	0.5063270	0.1185533	6
M-MMPA-GM		**0.3321563**	**0.2510415**	**0.2210539**	**0.0258682**	**1**		0.7925553	0.6853303	0.5615090	0.0707830	7
M-MMPA-NMM		0.3867489	0.2776597	0.1798777	0.0545513	2		1.1912621	0.8341616	0.5095818	0.1847302	9
SMPSO**[77]**		0.4149607	0.3095347	0.2291605	0.0506486	4		0.7133352	0.5479757	0.4820545	0.0574491	3
GDE3**[78]**		0.5201694	0.4075351	0.3007549	0.0677898	6		0.8095445	0.6578156	0.5194600	0.0763240	2
NSGA-II **[75]**		0.4991075	0.4273555	0.3187333	0.0409465	7		0.6311829	0.5526148	0.4933325	0.0409298	4
NSGA-III**[76]**		0.9749956	0.7088897	0.3982735	0.1297461	8		0.8316588	0.6906318	0.4245765	0.0950612	8
CGD**[79]**		1.1150704	0.9103322	0.8139913	0.0730449	9		**0.5519224**	**0.4128172**	**0.3273450**	**0.0586946**	**1**
MMPA	**F11**	1.0541008	0.9140664	0.7473422	0.0873847	6	**F16**	0.7966157	0.7148372	0.5887082	0.0582623	8
M-MMPA		1.1242998	0.9135275	0.7797280	0.0855297	2		0.9597111	0.6984640	0.5438246	0.0860789	7
M-MMPA-GM		1.3326784	1.0501452	0.8983201	0.1307061	8		0.8283786	0.6811610	0.5534130	0.0739292	6
M-MMPA-NMM		**1.0673929**	**0.8710884**	0.7714078	**0.0812600**	**1**		1.2606127	0.7494438	0.5824016	0.1586151	9
SMPSO**[77]**		1.3854619	0.9223705	0.7350342	0.1303490	7		0.7950558	0.5133948	0.4301140	0.0865700	2
GDE3**[78]**		1.1149688	0.9026249	0.5990922	0.1551085	4		0.8531279	0.6930558	0.5920599	0.0789791	4
NSGA-II **[75]**		1.4015981	1.0602505	0.8683648	0.1439125	9		0.5876563	0.5211501	0.4430568	0.0379918	3
NSGA-III**[76]**		1.1769564	0.9133178	0.7026231	0.1174313	5		0.7374603	0.6537289	0.5871291	0.0416890	5
CGD**[79]**		1.0741244	0.8714800	**0.5080352**	0.1499998	3		**0.5070428**	**0.4190443**	**0.3557101**	**0.0361707**	1

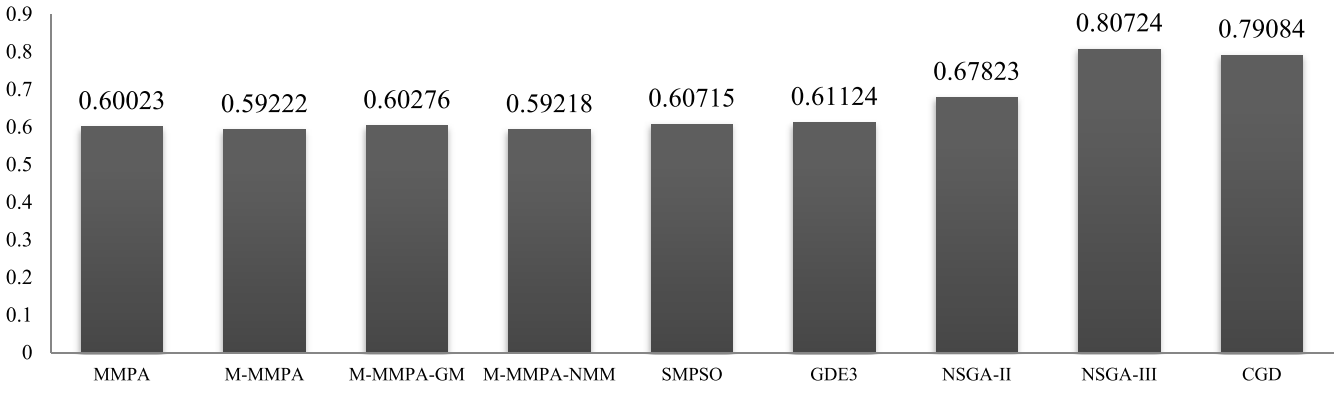
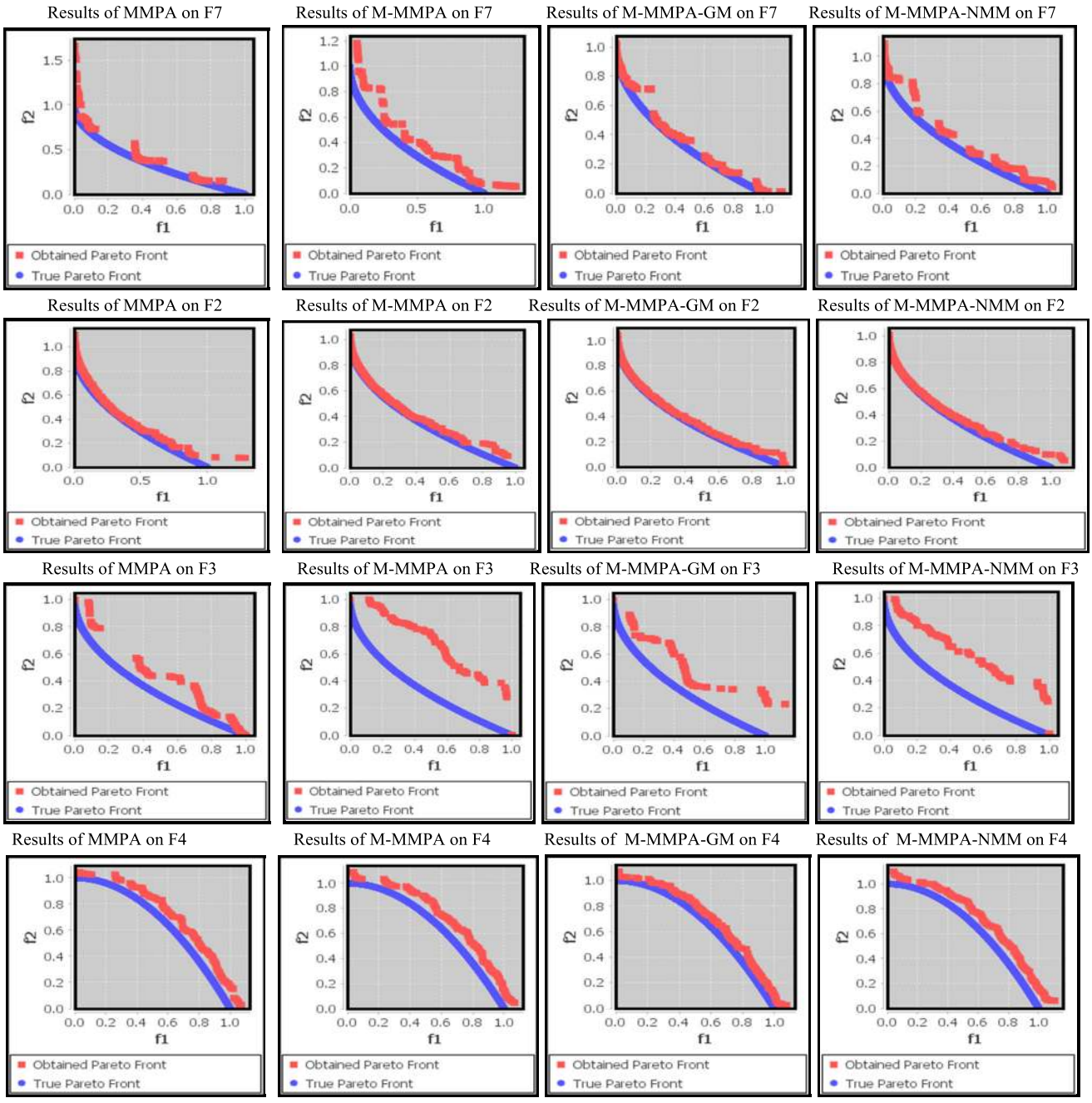


FIGURE 6. The average of the GSM values on CEC 2009 problems (F7-F16).

our proposed algorithm. Those three algorithms are compared with the proposed according to their implementation in JMetal frameworks under the cited parameters values. Broadly speaking, those three algorithms are executed 20 independent trials and the average IGD on each GLT test function is calculated and presented in Table 12, which elaborates the superiority of M-MMPA-GM for all test problems but F5, which could be solved better using DMOPSO.

The average of IGD values obtained by all the compared algorithms within 20 independent runs on GLT test problems



a. Pareto optimal front obtained by proposed algorithms

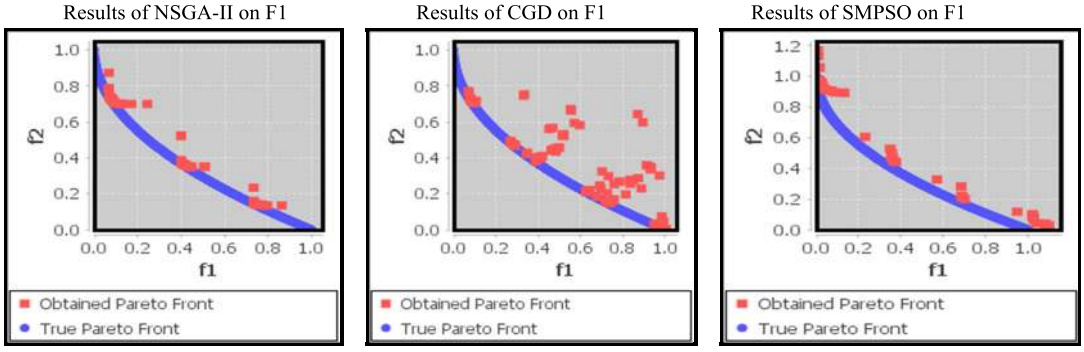
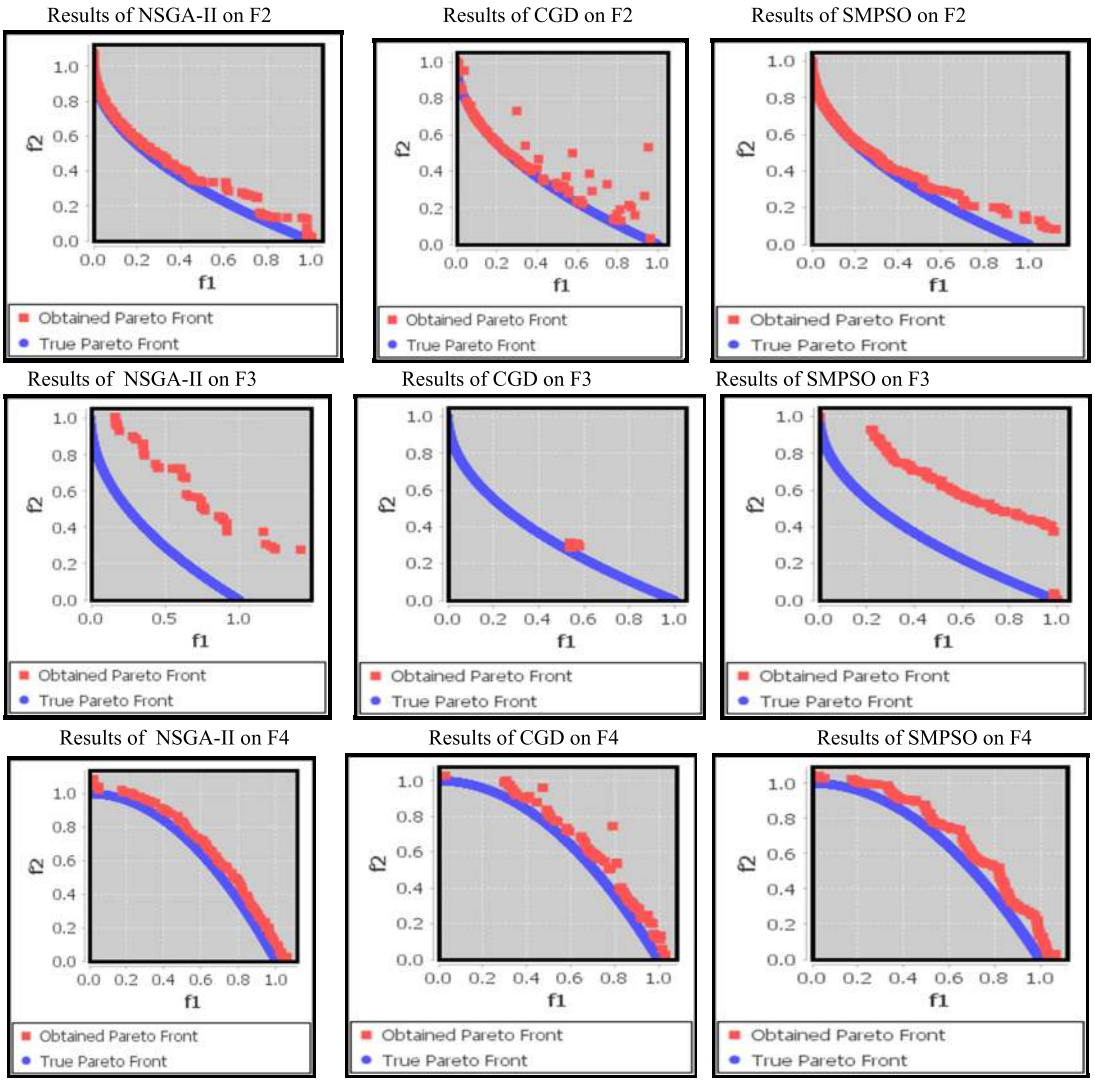


FIGURE 7. **Pareto optimal front obtained by all the algorithms.**



b. Pareto optimal front obtained by CGD, NSGA-II, and SMPSO algorithms

FIGURE 7. ***(Continued.)*** **Pareto optimal front obtained by all the algorithms.**

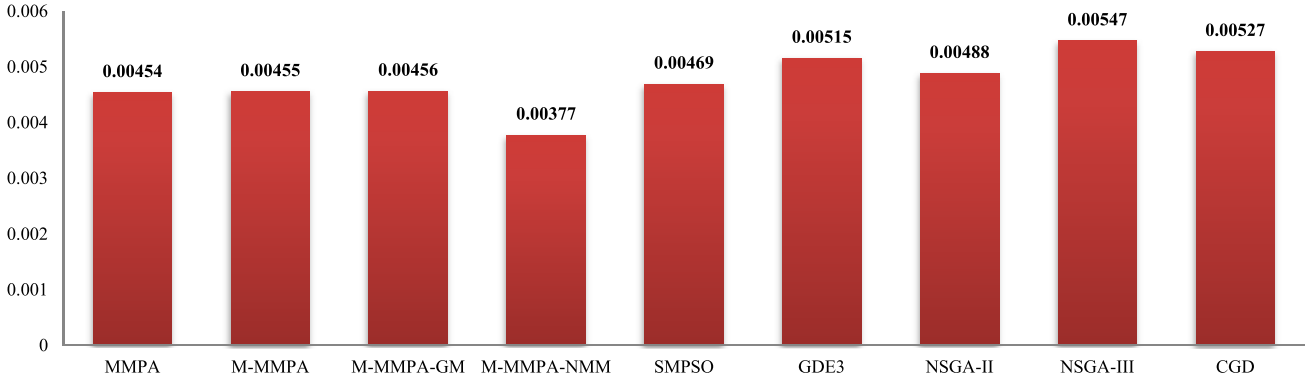


Fig. 8 The average of the IGD values on CEC 2020 problems (F17-F32)

FIGURE 8. **The average of the IGD values on CEC 2020 problems (F17-F32).**

is shown in Fig. 11 from which it is obvious that M-MMPA-GM is the best with a value of 0.00088, while MIBEA is worst with a value of 0.0124.

Besides, all algorithms used in our experiments are compared in Fig.12 in term of computational cost to show the speedup of the different algorithms. In general, all algorithms

TABLE 10. IGD values obtained by the proposed algorithm from F17 to F32.

Algorithm	F	Best	Avg	worst	Std	R	F	Best	Avg	Worst	Std	R
MMPA	**F17**	**0.0002056**	**0.0002147**	**0.0002214**	0.0000037	**1**	**F25**	0.0263042	0.0263058	0.0263075	0.0000010	8
M-MMPA		0.0002087	0.0002168	0.0002260	0.0000040	2		0.0262786	0.0262921	0.0263036	0.0000059	7
M-MMPA-GM		0.0002101	0.0002188	0.0002359	0.0000060	3		0.0262642	0.0262831	0.0262995	0.0000090	6
M-MMPA-NMM		0.0002102	0.0002213	0.0002281	**0.0000043**	5		**0.0103490**	**0.0136874**	**0.0160298**	0.0018618	**1**
SMPSO[77]		0.0002129	0.0002195	0.0002262	0.0000032	4		0.0263034	0.0263061	0.0263078	**0.0000011**	9
GDE3[78]		0.0002518	0.0002688	0.0002859	0.0000104	6		0.0111927	0.0180293	0.0263379	0.0074998	2
NSGA-II [75]		0.0002998	0.0003246	0.0003682	0.0000150	7		0.0112096	0.0240543	0.0263321	0.0053918	5
NSGA-III[76]		0.0007301	0.0008557	0.0010295	0.0000798	9		0.0114630	0.0227486	0.0266234	0.0064533	4
CGD[79]		0.0003084	0.0004319	0.0005419	0.0000684	8		0.0112658	0.0222996	0.0263506	0.0063472	3
MMPA	**F18**	0.0002226	0.0002320	0.0002575	0.0000105	3	**F26**	0.0014981	0.0016349	0.0017986	0.0000710	4
M-MMPA		**0.0002136**	**0.0002259**	**0.0002547**	0.0000108	**1**		**0.0014057**	**0.0015638**	**0.0017234**	0.0000823	**1**
M-MMPA-GM		0.0002137	0.0002289	0.0002553	0.0000107	2		0.0014457	0.0015858	0.0017363	0.0000765	3
M-MMPA-NMM		0.0002354	0.0002902	0.0004174	0.0000477	5		0.0015316	0.0017617	0.0020461	0.0001240	5
SMPSO[77]		0.0002260	0.0002329	0.0002482	**0.0000052**	4		0.0014468	0.0015646	0.0016824	**0.0000641**	2
GDE3[78]		0.0002524	0.0002649	0.0002845	0.0000077	6		0.0019221	0.0022495	0.0031580	0.0002523	7
NSGA-II [75]		0.0003081	0.0004690	0.0016988	0.0003937	7		0.0019752	0.0021920	0.0027002	0.0001691	6
NSGA-III[76]		0.0007470	0.0009895	0.0019982	0.0003367	9		0.0031956	0.0053396	0.0088839	0.0013313	9
CGD[79]		0.0002996	0.0005196	0.0017542	0.0003263	8		0.0029134	0.0033069	0.0041771	0.0004115	8
MMPA	**F19**	**0.0002099**	**0.0002183**	**0.0002304**	0.0000045	**1**	**F27**	0.0126409	0.0126500	0.0126574	0.0000048	5
M-MMPA		0.0002137	0.0002186	0.0002227	**0.0000025**	2		0.0126389	0.0126469	0.0126540	0.0000043	3
M-MMPA-GM		0.0002142	0.0002218	0.0002300	0.0000042	3		0.0126376	0.0126446	0.0126509	0.0000037	2
M-MMPA-NMM		0.0003922	0.0005360	0.0007064	0.0000801	6		**0.0056140**	**0.0086167**	**0.0126412**	0.0023097	**1**
SMPSO[77]		0.0002756	0.0003390	0.0005308	0.0000524	5		0.0126399	0.0126466	0.0126523	**0.0000033**	4
GDE3[78]		0.0002503	0.0002635	0.0002853	0.0000093	4		0.0126727	0.0127099	0.0128221	0.0000356	7
NSGA-II [75]		0.0003657	0.0006441	0.0024270	0.0006338	7		0.0126596	0.0126847	0.0127072	0.0000136	6
NSGA-III[76]		0.0010057	0.0030681	0.0219186	0.0048663	8		0.0128269	0.0131169	0.0135817	0.0002161	9
CGD[79]		0.0003619	0.0061083	0.0298783	0.0081109	9		0.0125810	0.0127748	0.0129059	0.0000714	8
MMPA	**F20**	**0.0002019**	0.0002165	0.0002357	0.0000100	2	**F28**	**0.0002764**	**0.0002871**	0.0003052	0.0000078	**1**
SMPSO[77]		0.0002039	0.0002175	0.0002386	0.0000086	3		0.0002763	0.0002972	0.0003105	0.0000085	4
M-MMPA		**0.0002019**	**0.0002127**	**0.0002328**	**0.0000096**	**1**		0.0002770	0.0002899	**0.0003020**	0.0000079	3
M-MMPA-GM		0.0001988	0.0002187	0.0002404	0.0000105	4		0.0002789	0.0002885	0.0003052	**0.0000063**	2
M-MMPA-NMM		0.0001926	0.0002239	0.0002407	0.0000111	5		0.0002843	0.0002992	0.0003370	0.0000126	5
GDE3[78]		0.0002729	0.0002895	0.0003419	0.0000159	6		0.0003320	0.0003818	0.0005003	0.0000331	6
NSGA-II [75]		0.0002827	0.0003087	0.0003387	0.0000159	7		0.0003701	0.0003965	0.0004335	0.0000179	7
NSGA-III[76]		0.0004697	0.0005625	0.0007011	0.0000624	8		0.0006257	0.0008858	0.0013837	0.0001926	9
CGD[79]		0.0004545	0.0007154	0.0008888	0.0000926	9		0.0006560	0.0008046	0.0009817	0.0000755	8
MMPA	**F21**	0.0004597	**0.0004896**	**0.0005068**	**0.0000113**	**1**	**F29**	0.0063712	0.0068901	0.0073241	0.0002867	2
M-MMPA		0.0004746	0.0004942	0.0005193	0.0000125	3		0.0066202	0.0069952	0.0074742	**0.0002433**	5
M-MMPA-GM		**0.0004588**	0.0004965	0.0005262	0.0000148	4		0.0064806	0.0070291	0.0075143	0.0002387	6
M-MMPA-NMM		0.0004695	0.0004917	0.0005334	0.0000131	2		0.0062941	0.0068041	0.0072604	0.0002795	3
SMPSO[77]		0.0004671	0.0005025	0.0005302	0.0000154	5		0.0066473	0.0076622	0.0100811	0.0007562	7
GDE3[78]		0.0005341	0.0006020	0.0006464	0.0000317	6		0.0074997	0.0086896	0.0110384	0.0010694	8
NSGA-II [75]		0.0006979	0.0007630	0.0008385	0.0000382	8		0.0069765	0.0079185	0.0097967	0.0007686	9
NSGA-III[76]		0.0010096	0.0013450	0.0021009	0.0002184	9		0.0065633	0.0069665	0.0076885	0.0003804	4
CGD[79]		0.0006804	0.0007618	0.0009095	0.0000594	7		**0.0061612**	**0.0065870**	**0.0071050**	0.0002906	**1**
1MMPA	**F22**	**0.0002095**	**0.0002137**	**0.0002183**	**0.0000020**	**1**	**F30**	0.0079027	0.0081580	0.0085094	0.0001839	4
M-MMPA		0.0002114	0.0002154	0.0002203	0.0000020	2		0.0077014	0.0082634	0.0088264	0.0002382	6
M-MMPA-GM		0.0002128	0.0002161	0.0002224	0.0000023	3		0.0079895	0.0082456	0.0085788	0.0001702	5
M-MMPA-NMM		0.0002113	0.0002165	0.0002249	0.0000031	4		0.0078219	0.0080647	0.0083013	**0.0001469**	2
SMPSO[77]		0.0002134	0.0002172	0.0002226	0.0000024	5		0.0079066	0.0088146	0.0107847	0.0007496	9
GDE3[78]		0.0002675	0.0002780	0.0002981	0.0000092	6		0.0074766	0.0088053	0.0106996	0.0009936	8
NSGA-II [75]		0.0002850	0.0003029	0.0003271	0.0000103	7		0.0080929	0.0086720	0.0099275	0.0004096	7
NSGA-III[76]		0.0005107	0.0006788	0.0009174	0.0001035	9		0.0076808	0.0081862	0.0088446	0.0003255	3
CGD[79]		0.0003247	0.0004060	0.0004683	0.0000342	8		**0.0071152**	**0.0074513**	**0.0081130**	0.0002053	**1**
MMPA	**F23**	**0.0002240**	**0.0002298**	**0.0002339**	**0.0000026**	**1**	**F31**	0.0073857	0.0077120	0.0082739	0.0002542	5
M-MMPA		0.0002278	0.0002315	0.0002364	0.0000022	2		0.0073582	0.0076862	0.0079899	0.0002122	4
M-MMPA-GM		0.0002257	0.0002316	0.0002368	0.0000033	3		0.0071091	0.0077921	0.0081809	0.0002689	6
M-MMPA-NMM		0.0002263	0.0002466	0.0003361	0.0000251	5		0.0066733	0.0074468	0.0080229	0.0003093	3
SMPSO[77]		0.0002334	0.0002381	0.0002588	0.0000055	4		0.0071111	0.0081244	0.0096778	0.0006506	9
GDE3[78]		0.0002776	0.0003013	0.0003427	0.0000179	6		0.0070933	0.0078416	0.0096449	0.0005512	7
NSGA-II [75]		0.0003005	0.0003308	0.0004265	0.0000295	7		0.0073548	0.0079061	0.0083952	0.0002737	8
NSGA-III[76]		0.0005031	0.0007174	0.0019631	0.0002996	8		0.0070873	0.0075289	0.0083895	0.0003462	2
CGD[79]		0.0010567	0.0011955	0.0015096	0.0000869	9		**0.0060201**	**0.0063764**	**0.0068412**	**0.0001965**	**1**
MMPA	**F24**	**0.0009910**	**0.0010908**	**0.0012042**	**0.0000558**	**1**	**F32**	0.0057943	0.0060551	0.0069351	0.0002431	3
M-MMPA		0.0010596	0.0011365	0.0012832	0.0000590	3		0.0058171	0.0061540	0.0064454	0.0001683	5
M-MMPA-GM		0.0010635	0.0011523	0.0012650	0.0000435	4		0.0057814	0.0061336	0.0065042	0.0001669	4
M-MMPA-NMM		0.0032361	0.0056385	0.0108606	0.0022041	6		0.0055634	0.0058465	0.0060234	**0.0001231**	2
SMPSO[77]		0.0010091	0.0011082	0.0012141	0.0000563	2		0.0058976	0.0065218	0.0073594	0.0003919	9
GDE3[78]		0.0013720	0.0150337	0.0223725	0.0099151	9		0.0059482	0.0064527	0.0076325	0.0004431	8
NSGA-II [75]		0.0013242	0.0047070	0.0223247	0.0073987	5		0.0059685	0.0064189	0.0076326	0.0003562	7
NSGA-III[76]		0.0032270	0.0082503	0.0227320	0.0072054	7		0.0056868	0.0062244	0.0069589	0.0003568	6
CGD[79]		0.0019880	0.0093592	0.0223874	0.0095238	8		**0.0049350**	**0.0052802**	**0.0055815**	0.0001715	**1**

were executed 20 independent runs on each problem of the GLT problems and the CPU time for those runs are computed and the average of this CPU time on all problems is presented in Fig.12. Inspecting this figure shows that M-MMPA-GM could occupy the fifth rank after DMOPSO, SMPSO, CGD, and GDE3, but this deterioration in time could be neglected

TABLE 11. GSM values obtained by algorithms on the test functions from F17 to F32.

Algorithm	F	Worst	Avg	Best	Std	R	F	Worst	Avg	Best	Std	
MMPA	F17	**0.1323014**	**0.0850549**	0.0644430	**0.0159709**	**1**	**F25**	0.7391431	0.7177257	0.7044218	**0.0094940**	**3**
M-MMPA		1.0122382	0.2207781	**0.0600270**	0.2573589	3		0.7441690	0.7269519	0.7163011	0.0073581	4
M-MMPA-GM		1.0154805	0.2623356	0.0705218	0.3063642	4		0.7552376	0.7331052	0.7081302	0.0110742	5
M-MMPA-NMM		1.0938403	0.4167580	0.0979562	0.3567811	7		1.4569580	1.0444828	0.7790375	0.1630250	9
SMPSO[77]		0.1306290	0.1041682	0.0818981	0.0136685	2		**0.5444379**	**0.5025050**	**0.4566501**	0.0204501	**1**
GDE3[78]		1.1050968	0.3375759	0.1984054	0.1873184	5		0.8304002	0.6985833	0.5752926	0.0969737	2
NSGA-II [75]		0.4304387	0.3704185	0.3146494	0.0348599	6		0.8865856	0.7883751	0.5267029	0.0753053	6
NSGA-III[76]		1.3408288	0.8668261	0.6766368	0.1369290	8		1.0577990	0.9761730	0.8774637	0.0464182	7
CGD[79]		1.4457194	1.0415161	0.5283105	0.2432564	9		1.0887446	0.9767895	0.7443333	0.0699926	8
MMPA	F18	0.2936579	**0.1262069**	**0.0772486**	**0.0442474**	**1**	**F26**	0.6200732	0.5881578	0.5497137	0.0206430	3
M-MMPA		1.9645798	0.3397389	0.0714629	0.5034531	4		0.5807981	0.5291226	0.4953193	0.0208850	2
M-MMPA-GM		1.8322820	0.3806309	0.0736476	0.4685124	5		**0.5489223**	**0.5262371**	**0.4902968**	0.0153890	1
M-MMPA-NMM		1.9010761	0.6175010	0.1358547	0.6263172	7		1.0323930	0.7340170	0.5432286	0.1447980	5
SMPSO[77]		**0.1571245**	0.1340782	0.1146338	0.0146358	2		0.7426094	0.7238100	0.7027659	**0.0103130**	**4**
GDE3[78]		0.6158171	0.3017741	0.2034412	0.0888234	3		0.7177457	0.6111528	0.5117690	0.0512895	6
NSGA-II [75]		0.5040458	0.3994215	0.3319078	0.0451933	6		0.7165940	0.6260376	0.5543400	0.0430662	7
NSGA-III[76]		1.0189294	0.8343272	0.7201201	0.0857976	8		1.0257150	0.9701188	0.8381889	0.0437217	9
CGD[79]		1.2733160	0.9574778	0.7908596	0.1211484	9		1.0831984	0.9596381	0.7224598	0.0876234	8
MMPA	F19	0.1496725	0.1080943	0.0717685	0.0187104	2	**F27**	0.7225325	0.6873786	0.6647209	0.0154603	4
M-MMPA		**0.1190339**	**0.0930405**	**0.0710350**	**0.0135292**	**1**		0.6796359	0.6523277	0.6239793	0.0143926	3
M-MMPA-GM		0.1547817	0.1135153	0.0811150	0.0183688	3		0.6710844	0.6497960	0.6222713	**0.0110349**	**2**
M-MMPA-NMM		1.7845330	1.4083414	0.5767761	0.3083325	9		1.0552947	0.7949206	0.6112507	0.1307827	5
SMPSO[77]		0.3853567	0.2502499	0.2009140	0.0423487	4		**0.5623603**	**0.5273415**	**0.4799266**	0.0199736	**1**
GDE3[78]		0.3665435	0.2809242	0.2237381	0.0385619	5		0.7862483	0.7326980	0.6682100	0.0312678	6
NSGA-II [75]		0.4864969	0.3848008	0.3137383	0.0463427	6		0.8379182	0.7594385	0.6890234	0.0358039	7
NSGA-III[76]		1.4902738	1.1423447	0.8491388	0.1631627	8		1.0747915	0.9438218	0.8091096	0.0685059	8
CGD[79]		1.0397979	0.8656659	0.5501532	0.1170972	7		1.0533105	0.9816558	0.8886348	0.0504335	9
MMPA	F20	1.1284827	0.1177173	**0.0371013**	0.2326008	4	**F28**	0.2184948	0.1927335	0.1630110	0.0141248	4
M-MMPA		0.8347696	0.1049244	0.0427644	0.1695917	3		0.2191796	0.1824043	0.1557526	0.0153288	3
M-MMPA-GM		**0.1159705**	**0.0712714**	0.0433636	0.0164211	**1**		0.2027897	0.1792033	0.1443346	0.0132733	2
M-MMPA-NMM		0.9888581	0.3603320	0.0740569	0.3026904	6		0.7688672	0.4016363	0.1886024	0.1815590	6
SMPSO[77]		0.1007519	0.0840592	0.0659771	**0.0100778**	2		**0.1920954**	**0.1699810**	**0.1433143**	**0.0160596**	**1**
GDE3[78]		0.4050595	0.3086152	0.2200393	0.0447300	5		0.4638215	0.3787746	0.3175524	0.0344629	5
NSGA-II [75]		1.4120785	0.4018863	0.2922128	0.2340801	7		0.5415552	0.4342524	0.3153524	0.0457761	7
NSGA-III[76]		1.3380825	0.7336213	0.5473672	0.1583313	9		1.1111240	0.9748758	0.9135554	0.0444746	8
CGD[79]		0.9576751	0.6650264	0.4211742	0.1259040	8		1.1773796	1.0408659	0.7876426	0.0869858	9
MMPA	F21	**0.1527503**	**0.1108217**	0.0786860	**0.0185625**	**1**	**F29**	0.6448435	0.4977999	0.4134791	0.0514318	5
M-MMPA		0.6803435	0.2189639	0.0893238	0.1663780	3		0.5893262	0.5019385	0.4259998	0.0436822	7
M-MMPA-GM		0.6280939	0.1623544	**0.0709200**	0.1223350	2		0.5645969	0.5016108	0.4352487	0.0314528	6
M-MMPA-NMM		0.6319503	0.2228451	0.1058647	0.1456514	4		0.5708720	0.4781727	0.3911366	0.0445857	2
SMPSO[77]		0.7026405	0.3202243	0.1119483	0.2213542	5		0.5509452	0.4814407	0.4204592	0.0419572	3
GDE3[78]		0.7447230	0.3227362	0.2443320	0.1168641	6		0.5977259	0.5426555	0.4228001	0.0408597	9
NSGA-II [75]		0.5591677	0.4461190	0.3832258	0.0481014	7		0.6168150	0.5415918	0.4513614	0.0430421	8
NSGA-III[76]		1.0157850	0.9274078	0.8179571	0.0552023	9		0.5862903	0.4885538	0.3662970	0.0528137	4
CGD[79]		1.2796894	0.8941872	0.4492199	0.2016959	8		**0.4373572**	**0.3590838**	**0.2768465**	**0.0334534**	**1**
MMPA	F22	**0.0987651**	**0.0663807**	**0.0395696**	0.0145029	**1**	**F30**	**0.5524383**	0.4996771	0.4673021	0.0225171	5
M-MMPA		0.1265000	0.0734068	0.0480079	0.0182781	2		0.5734352	0.4850341	0.4141420	0.0405445	2
M-MMPA-GM		0.3131316	0.0917199	0.0506446	0.0554523	4		0.5627039	0.4926717	0.4366251	**0.0306743**	4
M-MMPA-NMM		0.4011324	0.1379215	0.0583292	0.0901749	5		0.5677690	0.4871923	0.4296043	0.0345048	3
SMPSO[77]		0.0995645	0.0851977	0.0656716	**0.0092087**	3		0.5845323	0.5019651	0.4252516	0.0413090	6
GDE3[78]		0.3820285	0.2907710	0.2161441	0.0387720	6		0.5974949	0.5330868	0.4415268	0.0375298	7
NSGA-II [75]		0.4167643	0.3411870	0.2631644	0.0443604	7		0.6506346	0.5712224	0.5095361	0.0455259	8
NSGA-III[76]		0.8469693	0.7178725	0.5779975	0.0824130	8		0.6446352	0.5549860	0.4751198	0.0398688	8
CGD[79]		0.9849625	0.7827008	0.5199066	0.1175055	9		0.5572710	**0.4071977**	**0.3213741**	0.0557593	**1**
MMPA	F23	**0.1183226**	**0.0905541**	**0.0699940**	**0.0127922**	**1**	**F31**	0.6501059	0.5361133	0.4736276	0.0358985	6
M-MMPA		0.1288479	0.0962991	0.0655761	0.0171104	2		0.5926179	0.5299680	0.4278191	0.0420875	5
M-MMPA-GM		0.1366288	0.0987676	0.0705549	0.0177191	3		0.5817661	0.5198830	0.4586108	0.0371481	4
M-MMPA-NMM		1.8615991	1.0613188	0.1893099	0.5970238	9		0.6483112	0.5052395	0.4313791	0.0490956	2
SMPSO[77]		0.1376316	0.1112318	0.0771095	0.0168920	4		0.5831235	0.5149692	0.4683722	**0.0306189**	**3**
GDE3[78]		0.3572100	0.3121946	0.2490515	0.0316391	5		0.6521259	0.5712917	0.4887270	0.0395932	8
NSGA-II [75]		0.4396398	0.3615704	0.2677429	0.0412673	6		0.7519581	0.6025293	0.5080060	0.0585134	9
NSGA-III[76]		1.7236346	0.8053768	0.6773005	0.2343443	7		0.6594674	0.5265587	0.4145760	0.0559186	7
CGD[79]		1.0306880	0.9214491	0.7018229	0.0889637	8		**0.4214968**	**0.3570695**	**0.2938753**	0.0322248	**1**
MMPA	F24	**0.5286481**	**0.4962416**	**0.4535384**	0.0209441	**1**	**F32**	0.5925229	0.5271017	0.4715934	0.0330200	5
M-MMPA		0.5595505	0.5154133	0.4777570	0.0217494	3		0.5765519	0.5289674	0.4682443	0.0287613	6
M-MMPA-GM		0.5465653	0.5071629	0.4701259	0.0185698	2		0.5849707	0.5246570	0.4697245	0.0328902	4
M-MMPA-NMM		1.2917818	1.0471528	0.6916170	0.1596963	7		0.5651703	0.4937984	0.4287677	0.0407713	2
SMPSO[77]		0.7426094	0.7238100	0.7027659	**0.0103130**	6		0.6040475	0.5122742	0.4187430	0.0432497	3
GDE3[78]		0.7879414	0.6246035	0.5276991	0.0694672	4		0.6443344	0.5673075	0.4917918	0.0426379	8
NSGA-II [75]		0.7301970	0.6397306	0.5776821	0.0386479	5		0.6656440	0.5891531	0.5391816	0.0324797	9
NSGA-III[76]		1.0296681	0.9336629	0.7870947	0.0719072	8		0.6438202	0.5465739	0.4706734	0.0510737	7
CGD[79]		1.1489197	1.0101841	0.8241713	0.0837227	9		**0.4946730**	**0.3462167**	**0.2937877**	**0.0405450**	**1**

with the superiority of M-MMPA-GM in terms of IGD and GSM, which makes it a strong alternative for tackling the multiobjective optimization problems

Finally, the T-test as a statistical significance [85] is used to show the difference between the IGD obtained by the best five compared algorithms in our comparison with the

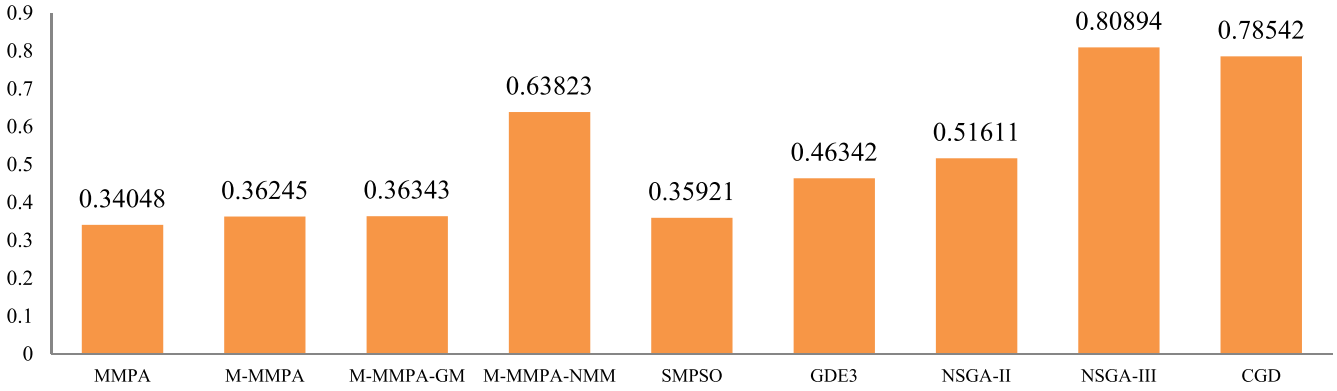


FIGURE 9. The average of *GSM* values on CEC 2020 problems (F17-F32).

TABLE 12. IGD values of the objective space obtained for GLT test problems.

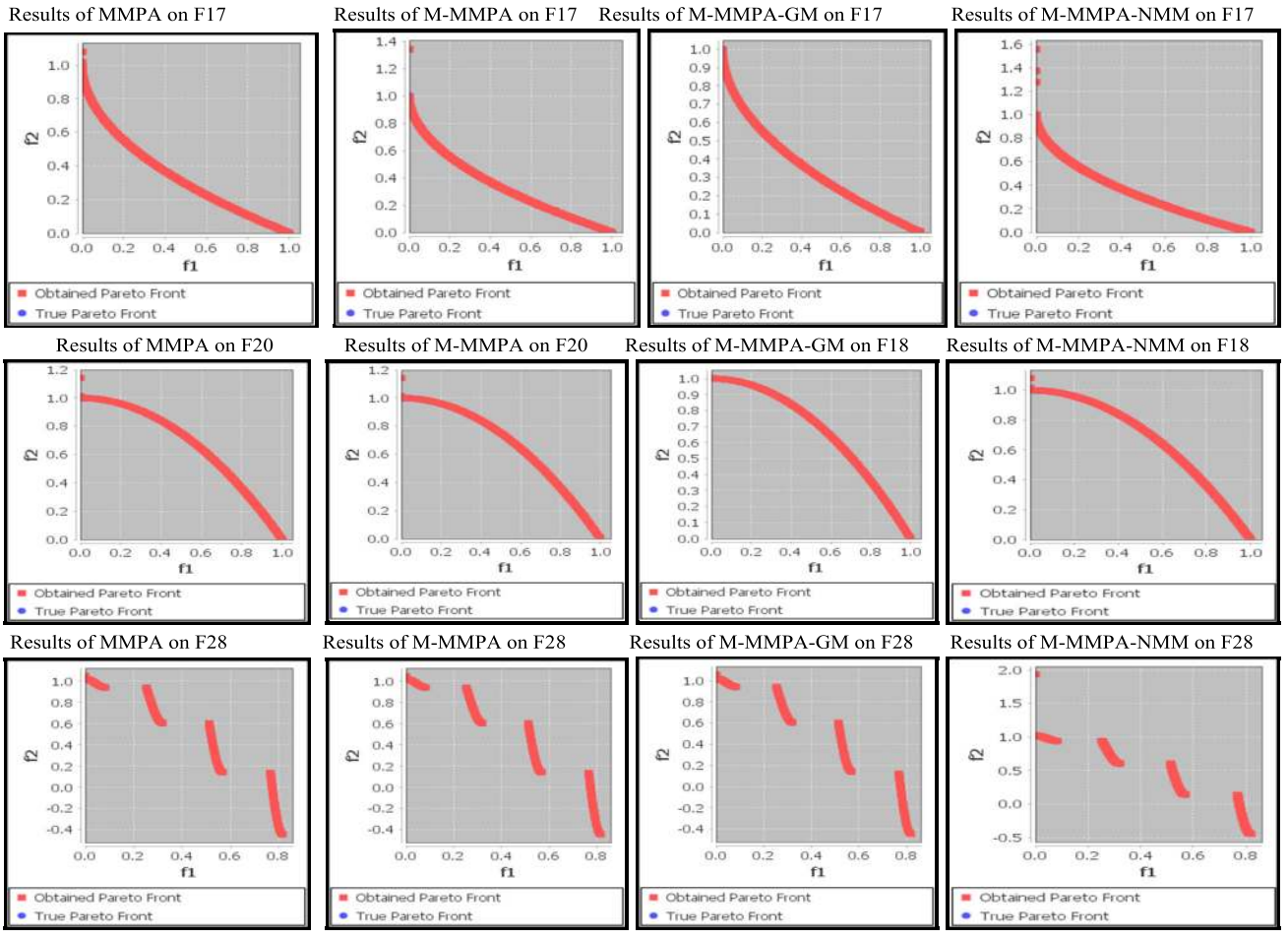
Algorithm	F	Best	Avg	Worst	Std	R	F	Best	Avg	Worst	Std	R
MMPA	**F1**	0.0001250	0.0010578	0.0032050	0.0009257	4	**F4**	0.0002137	0.0023502	0.0087560	0.0023488	4
M-MMPA		0.0001243	0.0001829	0.0003493	0.0000597	2		0.0002252	0.0006522	0.0044208	**0.0010283**	2
M-MMPA-GM		**0.0000809**	**0.00009673**	**0.0001103**	**0.0000155**	**1**		**0.0002021**	**0.0005067**	**0.0022453**	0.0021751	**1**
M-MMPA-NMM		0.0001232	0.0002264	0.0005901	0.0001086	3		0.0002163	0.0013843	0.0085426	0.0024431	3
DMOPSO[77]		0.0009548	0.0022426	0.0056539	0.0013006	5		0.0015938	0.0065047	0.0170835	0.0045052	5
MIBEA[78]		0.0043636	0.0054558	0.0061673	0.0006444	6		0.0041597	0.0120357	0.0172371	0.0036199	6
SMSEMOA		0.0055373	0.0070142	0.0083346	0.0006776	7		0.0106121	0.0143957	0.0216406	0.0029460	7
MMPA	**F2**	0.0012466	0.0012839	0.0013565	0.0000264	2	**F5**	0.0017768	0.0020398	0.0023382	0.0001605	5
M-MMPA		0.0012358	0.0013173	0.0014435	0.0000467	4		0.0016279	0.0019113	0.0021303	0.0001303	3
M-MMPA-GM		**0.0011892**	**0.0012430**	**0.0013022**	**0.0000304**	**1**		0.0017022	0.0018757	0.0019940	0.0000938	**2**
M-MMPA-NMM		0.0012497	0.0013053	0.0014750	0.0000486	3		0.0017827	0.0019579	0.0023195	0.0001315	4
DMOPSO[77]		0.0105279	0.0182651	0.0341913	0.0055900	5		**0.0015919**	**0.0016734**	**0.0017556**	**0.0000449**	1
GDE3[78]		0.0184279	0.0380438	0.0862433	0.0187550	6		0.0052394	0.0077270	0.0135211	0.0016795	7
SMSEMOA		0.0206043	0.0425983	0.1018377	0.0172613	7		0.0048267	0.0059227	0.0077079	0.0006701	6
MMPA	**F3**	0.0001818	0.0002647	0.0010118	0.0001807	4	**F6**	0.0012812	0.0014900	0.0016440	0.0000949	3
M-MMPA		0.0001853	0.0002446	0.0003900	0.0000664	3		0.0013489	0.0015204	0.0017451	0.0000845	5
M-MMPA-GM		**0.0001789**	**0.0002004**	**0.0004212**	**0.0000547**	**1**		**0.0012174**	**0.0013749**	**0.0015421**	**0.0000932**	**1**
M-MMPA-NMM		0.0001821	0.0002354	0.0008243	0.0001396	2		0.0012770	0.0014386	0.0017843	0.0001223	2
DMOPSO[77]		0.0011129	0.0026323	0.0044154	0.0009742	5		0.0013032	0.0015038	0.0017429	0.0001263	4
MIBEA[78]		0.0031219	0.0059346	0.0122894	0.0018118	7		0.0023228	0.0054275	0.0087606	0.0016363	6
SMSEMOA		0.0033684	0.0054009	0.0081503	0.0014711	6		0.0031929	0.0046757	0.0063207	0.0008848	5

TABLE 13. T-test results on GLT.

	CGD		*DMOPSO*		*GDE3*		*MIBEA*		*NSGAII*	
	h	*p*-value	h	*p*-value	h	h	h	*p*-value	h	*p*-value
F1	**0**	**4.37869E-01**	**1**	**1.44067E-06**	**1**	**4.32677E-03**	**1**	**1.50758E-21**	**0**	**7.95792E-17**
F2	**1**	**1.65917E-14**	**1**	**1.15046E-19**	**1**	**4.05207E-16**	**1**	**6.20049E-15**	**1**	**4.67883E-11**
F3	**1**	**8.96273E-14**	**1**	**2.35613E-13**	**1**	**3.95433E-10**	**1**	**3.99304E-23**	**1**	**3.62635E-15**
F4	**1**	**2.37340E-08**	**1**	**3.32703E-08**	**1**	**4.14454E-05**	**1**	**1.11732E-16**	**1**	**8.71753E-16**
F5	**1**	**6.45668E-05**	**1**	**8.56671E-07**	**1**	**1.70596E-11**	**1**	**1.71614E-17**	**1**	**8.26441E-12**
F6	**0**	**9.20556E-02**	**0**	**1.67989E-01**	**1**	**7.10353E-03**	**1**	**2.01529E-12**	**1**	**2.45490E-10**

proposed algorithm: M-MMPA-GM. This test is used to see the superiority of the proposed algorithm for GLT benchmark problems and the outcomes of this test is shown in Table 12, that show the acceptance of alternative hypothesis for F2, F3, F4, and F5 and this show the superiority of the proposed, while accepting the Null hypothesis for F1 and F6 with CGD and DMOPSO outcomes

Since CEC 2020 is multi-modal multi-objective, and the MMPA could achieve good solutions on this benchmark compared with the other algorithms, MMPA is considered the



a. Pareto optimal front obtained by proposed algorithms

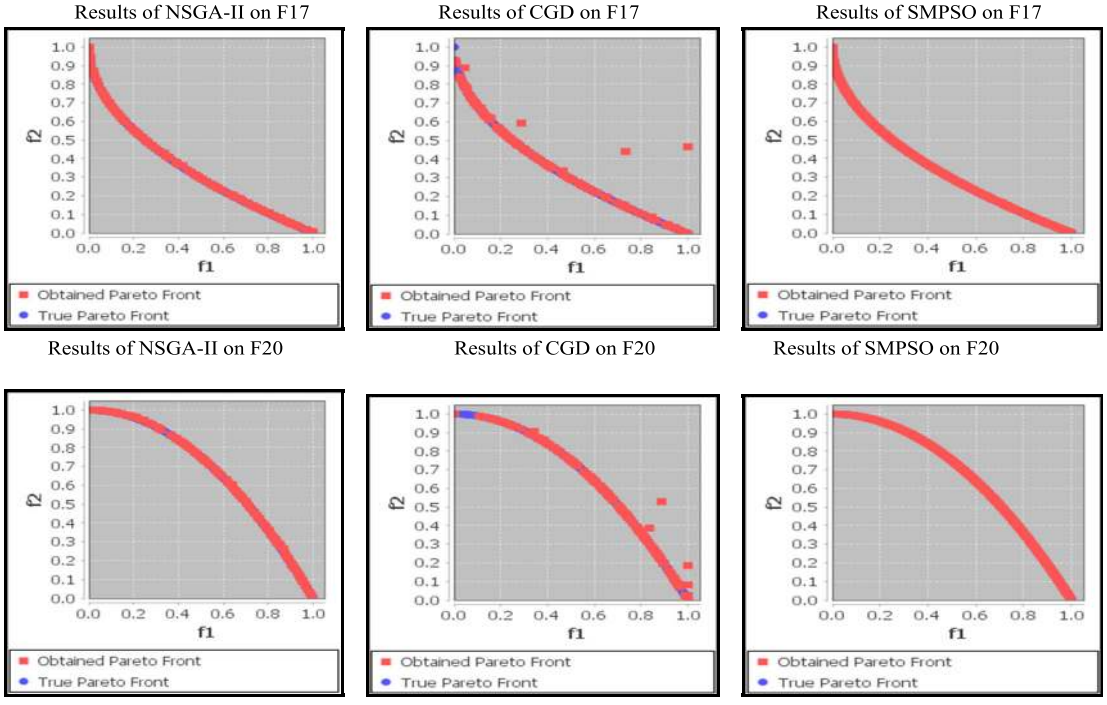
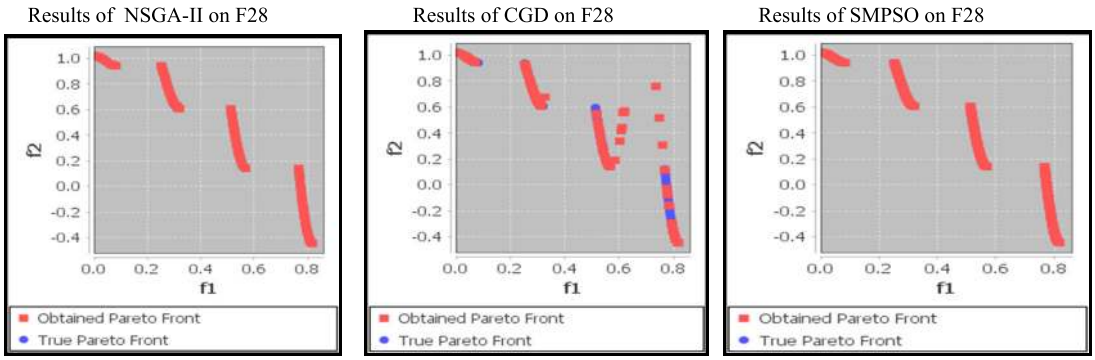


FIGURE 10. Pareto optimal front obtained by all the algorithms on CEC 2020.



b. Pareto optimal front obtained by CGD, NSGA-II, and SMPSO algorithms

FIGURE 10. ***(Continued.)*** **Pareto optimal front obtained by all the algorithms on CEC 2020.**

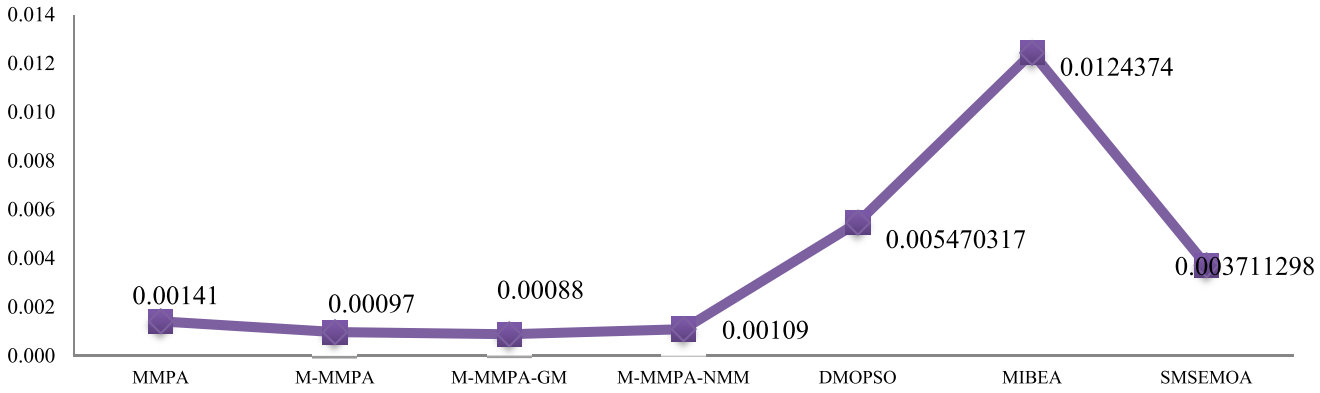


FIGURE 11. **The average of IGD values on GLT problems (F1-F6).**

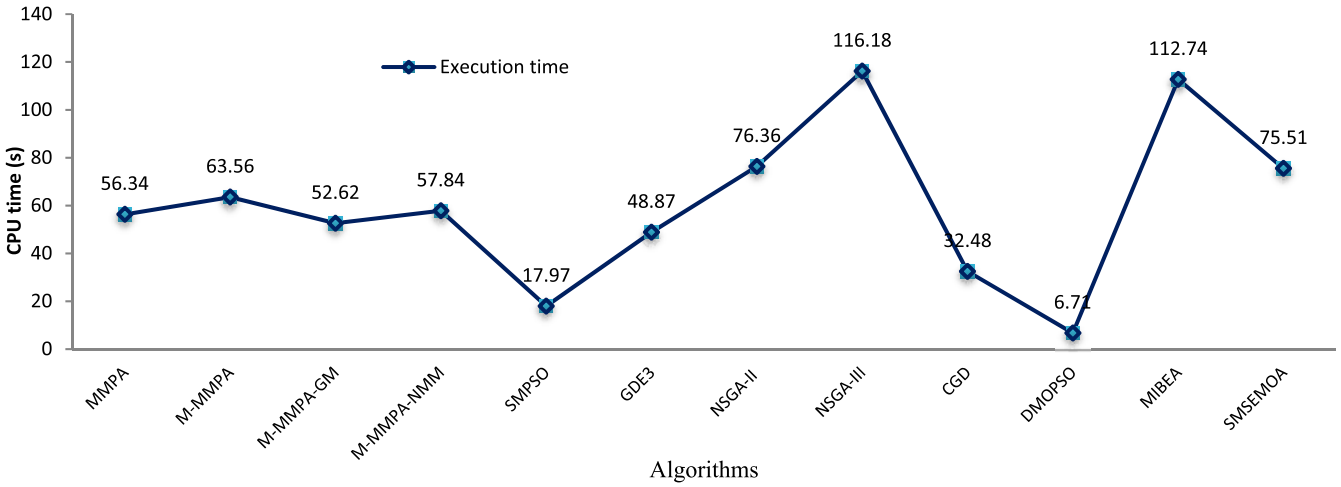


FIGURE 12. **Computational cost consumed by each algorithm on GLT problems.**

best approach for solving this type of problem. Meanwhile, M-MMPA-GM is the best for overcoming multi-objective problems. As a help to the DMs when solving the MMPA, it is preferred to use M-MMPA-GM when solving the multi-objective problem, while using MMPA when solving the multi-model, multi-objective problem.

VI. CONCLUSION AND FUTURE WORK

This paper proposes four novel versions of MPA to solve multi-objective (and multi-modal) optimization problems. The first is a multi-objective version of the standard MPA. The second is based on modifying the standard MPA using a novel dominance strategy based on

exploration-exploitation (DSEE) which counts the number of dominated solutions for each solution, and the solutions with the high dominance would have an exploitation phase, and the others with small dominance would have an exploration phase. The third version integrates MPA with a novel Gaussian-based strategy to explore more solutions around the best-so-far non-dominated solutions. The Gaussian-based strategy uses the methodology of exploration and exploitation—within a percentage predefined by the user, the small sigma value of Gaussian distribution can be used to exploit better solutions with a small distance of the current solution, and the remaining percentage of the sigma value is quite large to explore more solutions with large distances from the current solutions. The last version uses the Nelder-Mead simplex at the start of the optimization process to build up a front that will help MPA find better solutions within the optimization process. The performances of the four proposed versions are evaluated on CEC 2009, CEC 2020, and GLT problems and compared with CDG, SMPSO, NSGA-II, NSGA-III, and GDE3. The results show the superiority of the four proposed versions compared with the existing algorithms. Among the proposed algorithms, M-MMPA-GM could be superior for most test functions, where it could fulfill average IGD values of 0.00088, 0.01232, and 0.00456 for GLT, CEC 2009, and CEC 2020, respectively. It is worth mentioning that the proposed algorithms could be slightly competitive for CEC2020. For GSM metric, M-MMPA-GM could come true with average values of 0.60276, 0.36343, and 0.2768 for those three benchmarks mentioned above, respectively. Future work includes applying MPA for solving multi-dimensional knapsack problems, DNA fragment assembly problems, and shop scheduling problems. Moreover, also as our future work, we will apply those variants for tackling some of the real-world problems such as multiobjective DNA fragment assembly problems to minimize the number of contigs and maximize the overlap score, real-time task scheduling in multiprocessor systems, and task scheduling problem in Fog computing.

Conflict of interest

The authors declare that there is no conflict of interest in the research.

Funding

This research has no funding source.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

REFERENCES

[1] A. A. Mousa, M. A. El-Shorbagy, and W. F. Abd-El-Wahed, "Local search based hybrid particle swarm optimization algorithm for multiobjective optimization," *Swarm Evol. Comput.*, vol. 3, pp. 1–14, Apr. 2012.

[2] J. Branke, "Finding knees in multi-objective optimization," in *Proc. Int. Conf. Parallel Problem Solving Nature*. Berlin, Germany: Springer, 2004, pp. 722–731, doi: 10.1007/978-3-540-30217-9_73.

[3] A. K. Mishra, "An efficient Jaya algorithm for multi-objective permutation flow shop scheduling problem," in *Advanced Engineering Optimization Through Intelligent Techniques*. Singapore: Springer, 2020, pp. 113–125, doi: 10.1007/978-981-13-8196-6_11.

[4] H. Monsef, M. Naghashzadegan, A. Jamali, and R. Farmani, "Comparison of evolutionary multi objective optimization algorithms in optimum design of water distribution network," *Ain Shams Eng. J.*, vol. 10, no. 1, pp. 103–111, Mar. 2019.

[5] C. Wu, J. Wang, X. Chen, P. Du, and W. Yang, "A novel hybrid system based on multi-objective optimization for wind speed forecasting," *Renew. Energy*, vol. 146, pp. 149–165, Feb. 2020.

[6] D. Wang, L. Geng, Y.-J. Zhao, Y. Yang, Y. Huang, Y. Zhang, and H.-B. Shen, "Artificial intelligence-based multi-objective optimization protocol for protein structure refinement," *Bioinformatics*, vol. 36, no. 2, pp. 437–448, Jan. 2020, doi: 10.1093/bioinformatics/btz544.

[7] T. George and T. Amudha, "Genetic algorithm based multi-objective optimization framework to solve traveling salesman problem," in *Advances in Computing and Intelligent Systems*. Singapore: Springer, 2020, pp. 141–151, doi: 10.1007/978-981-15-0222-4_12.

[8] D. A. V. Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithms: Analyzing the state-of-the-art," *Evol. Comput.*, vol. 8, no. 2, pp. 125–147, Jun. 2000.

[9] N. Acevedo, C. Rey, C. Contreras-Bolton, and V. Parada, "Automatic design of specialized algorithms for the binary knapsack problem," *Expert Syst. Appl.*, vol. 141, Mar. 2020, Art. no. 112908.

[10] M. Abdel-Basset, D. El-Shahat, I. El-henawy, V. H. C. de Albuquerque, and S. Mirjalili, "A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection," *Expert Syst. Appl.*, vol. 139, Jan. 2020, Art. no. 112824.

[11] M. Habib, "Multi-objective particle swarm optimization for BotNet detection in Internet of Things," in *Evolutionary Machine Learning Techniques*. Singapore: Springer, 2020, pp. 203–229, doi: 10.1007/978-981-32-9990-0_10.

[12] A. Rajagopalan and D. R. R. Modale Senthilkumar, "Optimal scheduling of tasks in cloud computing using hybrid firefly-genetic algorithm," in *Advances in Decision Sciences, Image Processing, Security and Computer Vision*. Cham, Switzerland: Springer, 2020, pp. 678–687, doi: 10.1007/978-3-030-24318-0_77.

[13] T. Chugh, K. Sindhya, J. Hakanen, and K. Miettinen, "A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms," *Soft Comput.*, vol. 23, no. 9, pp. 3137–3166, May 2019.

[14] K. Deb, "Constrainced multi-objective evolutionary algorithm," in *Evolutionary and Swarm Intelligence Algorithms*. Cham, Switzerland: Springer, 2019, pp. 85–118, doi: 10.1007/978-3-319-91341-4_6.

[15] M. T. M. Emmerich and A. H. Deutz, "A tutorial on multiobjective optimization: Fundamentals and evolutionary methods," *Natural Comput.*, vol. 17, no. 3, pp. 585–609, Sep. 2018.

[16] A. Santiago, B. Dorronsoro, A. J. Nebro, J. J. Durillo, O. Castillo, and H. J. Fraire, "A novel multi-objective evolutionary algorithm with fuzzy logic based adaptive selection of operators: FAME," *Inf. Sci.*, vol. 471, pp. 233–251, Jan. 2019.

[17] S. Zapotecas-Martínez, A. López-Jaimes, and A. García-Nájera, "LIBEA: A lebesgue indicator-based evolutionary algorithm for multi-objective optimization," *Swarm Evol. Comput.*, vol. 44, pp. 404–419, Feb. 2019.

[18] Z.-Z. Liu and Y. Wang, "Handling constrained multiobjective optimization problems with constraints in both the decision and objective spaces," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 870–884, Oct. 2019.

[19] G. Chen and J. Li, "A diversity ranking based evolutionary algorithm for multi-objective and many-objective optimization," *Swarm Evol. Comput.*, vol. 48, pp. 274–287, Aug. 2019.

[20] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[21] Y. Tian, R. Cheng, X. Zhang, F. Cheng, and Y. Jin, "An indicator-based multiobjective evolutionary algorithm with reference point adaptation for better versatility," *IEEE Trans. Evol. Comput.*, vol. 22, no. 4, pp. 609–622, Aug. 2018.

[22] S. Seifollahi-Aghmiuni and O. B. Haddad, "Multi objective optimization with a new evolutionary algorithm," *Water Resour.Manage.*, vol. 32, no. 12, pp. 4013–4030, 2018.

[23] D. M. Pedroso, M. R. Bonyadi, and M. Gallagher, "Parallel evolutionary algorithm for single and multi-objective optimisation: Differential evolution and constraints handling," *Appl. Soft Comput.*, vol. 61, pp. 995–1012, Dec. 2017.

[24] R. Hernández Gómez and C. A. C. Coello, "A parallel version of SMS-EMOA for many-objective optimization problems," in *Proc. Int. Conf. Parallel Problem Solving Nature*. Cham, Switzerland: Springer, 2016, pp. 568–577, doi: 10.1007/978-3-319-45823-6_53.

[25] Q. Zhang, "Multiobjective optimization test instances for the CEC 2009 special session and competition," Special Session on Performance Assessment of Multi-Objective Optimization Algorithms, Univ. Essex, Colchester, U.K., and Nanyang Technol. Univ., Singapore, Tech. Rep. CES-887, 2008, p. 264.

[26] *Matlab, C++*. Accessed: Sep. 1, 2020. [Online]. Available: https://github.com/P-N-Suganthan/2020-Multimodal-Multi-Objective-Benchmark

[27] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, Jun. 2000.

[28] J. Knowles and D. Corne, "The Pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation," in *Proc. Congr. Evol. Comput. (CEC)*, Washington, DC, USA, 1999, doi: 10.1109/CEC.1999.781913.

[29] G. G. Yen and H. Lu, "Dynamic multiobjective evolutionary algorithm: Adaptive cell-based rank and density estimation," *IEEE Trans. Evol. Comput.*, vol. 7, no. 3, pp. 253–274, Jun. 2003.

[30] J. Liang, W. Xu, C. Yue, K. Yu, H. Song, O. D. Crisalle, and B. Qu, "Multimodal multiobjective optimization with differential evolution," *Swarm Evol. Comput.*, vol. 44, pp. 1028–1059, Feb. 2019.

[31] A. Atashpendar, B. Dorronsoro, G. Danoy, and P. Bouvry, "A scalable parallel cooperative coevolutionary PSO algorithm for multi-objective optimization," *J. Parallel Distrib. Comput.*, vol. 112, pp. 111–125, Feb. 2018.

[32] V. Mokarram and M. R. Banan, "A new PSO-based algorithm for multi-objective optimization with continuous and discrete design variables," *Structural Multidisciplinary Optim.*, vol. 57, no. 2, pp. 509–533, Feb. 2018.

[33] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 256–279, Jun. 2004.

[34] B. Y. Qu, J. J. Liang, and P. N. Suganthan, "Niching particle swarm optimization with local search for multi-modal optimization," *Inf. Sci.*, vol. 197, pp. 131–143, Aug. 2012.

[35] W. Zhang, G. Li, W. Zhang, J. Liang, and G. G. Yen, "A cluster based PSO with leader updating mechanism and ring-topology for multimodal multi-objective optimization," *Swarm Evol. Comput.*, vol. 50, Nov. 2019, Art. no. 100569.

[36] A. Askarzadeh, "A novel Metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm," *Comput. Struct.*, vol. 169, pp. 1–12, Jun. 2016.

[37] H. Nobahari and A. Bighashdel, "MOCSA: A multi-objective crow search algorithm for multi-objective optimization," in *Proc. 2nd Conf. Swarm Intell. Evol. Comput. (CSIEC)*, Mar. 2017, pp. 60–65.

[38] S. H. R. Pasandideh and S. Khalilpourazari, "Sine cosine crow search algorithm: A powerful hybrid meta heuristic for global optimization," 2018, *arXiv:1801.08485*. [Online]. Available: http://arxiv.org/abs/1801.08485

[39] M. A. Tawhid and V. Savsani, "Multi-objective sine-cosine algorithm (MO-SCA) for multi-objective engineering design problems," *Neural Comput. Appl.*, vol. 31, no. 2, pp. 915–929, 2019.

[40] M. A. Elaziz, L. Li, K. P. N. Jayasena, and S. Xiong, "Multiobjective big data optimization based on a hybrid salp swarm algorithm and differential evolution," *Appl. Math. Model.*, vol. 80, pp. 929–943, Apr. 2020.

[41] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017.

[42] G. Dhiman, "EMoSOA: A new evolutionary multi-objective seagull optimization algorithm for global optimization," *Int. J. Mach. Learn. Cybern.*, vol. 12, no. 2, pp. 571–596, 2020.

[43] S. Jiang and Z. Chen, "A two-phase evolutionary algorithm framework for multi-objective optimization," *Appl. Intell.*, 2020, doi: 10.1007/s10489-020-01988-7.

[44] Y. Tian, R. Liu, X. Zhang, H. Ma, K. C. Tan, and Y. Jin, "A multipopulation evolutionary algorithm for solving large-scale multi-modal multi-objective optimization problems," *IEEE Trans. Evol. Comput.*, early access, Dec. 15, 2020, doi: 10.1109/TEVC.2020.3044711.

[45] V. Palakonda and R. Mallipeddi, "An evolutionary algorithm for multi and many-objective optimization with adaptive mating and environmental selection," *IEEE Access*, vol. 8, pp. 82781–82796, 2020.

[46] Z. Wang, H. Li, and H. Yu, "MOEA/UE: A novel multi-objective evolutionary algorithm using a uniformly evolving scheme," *Neurocomputing*, to be published, doi: 10.1016/j.neucom.2020.04.149.

[47] X.-S. Yang, M. Karamanoglu, and X. He, "Flower pollination algorithm: A novel approach for multiobjective optimization," *Eng. Optim.*, vol. 46, no. 9, pp. 1222–1237, Sep. 2014.

[48] A. Panda and S. Pani, "A symbiotic organisms search algorithm with adaptive penalty function to solve multi-objective constrained optimization problems," *Appl. Soft Comput.*, vol. 46, pp. 344–360, Sep. 2016.

[49] X.-S. Yang, "Bat algorithm for multi-objective optimisation," 2012, *arXiv:1203.6571*. [Online]. Available: http://arxiv.org/abs/1203.6571

[50] S. Mirjalili, P. Jangir, and S. Saremi, "Multi-objective ant lion optimizer: A multi-objective optimization algorithm for solving engineering problems," *Appl. Intell.*, vol. 46, no. 1, pp. 79–95, 2017.

[51] M. A. El Aziz, A. A. Ewees, A. E. Hassanien, M. Mudhsh, and S. Xiong, "Multi-objective whale optimization algorithm for multilevel thresholding segmentation," in *Advances in Soft Computing and Machine Learning in Image Processing*. Cham, Switzerland: Springer, 2018, pp. 23–39, doi: 10.1007/978-3-319-63754-9_2.

[52] X. Lai, C. Li, N. Zhang, and J. Zhou, "A multi-objective artificial sheep algorithm," *Neural Comput. Appl.*, vol. 31, no. 8, pp. 4049–4083, Aug. 2019.

[53] Q. Lin, Q. Zhu, N. Wang, P. Huang, W. Wang, J. Chen, and Z. Ming, "A multi-objective immune algorithm with dynamic population strategy," *Swarm Evol. Comput.*, vol. 50, Nov. 2019, Art. no. 100477.

[54] V. Savsani and M. A. Tawhid, "Non-dominated sorting moth flame optimization (NS-MFO) for multi-objective problems," *Eng. Appl. Artif. Intell.*, vol. 63, pp. 20–32, Aug. 2017.

[55] Vikas and S. J. Nanda, "Multi-objective moth flame optimization," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2016, pp. 2470–2476.

[56] S. Mirjalili, S. Saremi, S. M. Mirjalili, and L. D. S. Coelho, "Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization," *Expert Syst. Appl.*, vol. 47, pp. 106–119, Apr. 2016.

[57] S. Z. Mirjalili, S. Mirjalili, S. Saremi, H. Faris, and I. Aljarah, "Grasshopper optimization algorithm for multi-objective optimization problems," *Int. J. Speech Technol.*, vol. 48, no. 4, pp. 805–820, Apr. 2018.

[58] A. Tharwat, E. H. Houssein, M. M. Ahmed, A. E. Hassanien, and T. Gabel, "MOGOA algorithm for constrained and unconstrained multi-objective optimization problems," *Int. J. Speech Technol.*, vol. 48, no. 8, pp. 2268–2283, Aug. 2018.

[59] M. Abdel-Basset, R. Mohamed, and M. Abouhawwash, "Balanced multi-objective optimization algorithm using improvement based reference points approach," *Swarm Evol. Comput.*, vol. 60, Feb. 2021, Art. no. 100791.

[60] M. Abdel-Basset, R. Mohamed, and S. Mirjalili, "A novel whale optimization algorithm integrated with Nelder–Mead simplex for multi-objective optimization problems," *Knowl.-Based Syst.*, vol. 212, Jan. 2021, Art. no. 106619.

[61] H. Seada and K. Deb, "U-NSGA-III: A unified evolutionary optimization procedure for single, multiple, and many objectives: Proof-of-principle results," in *Proc. Int. Conf. Evol. Multi-Criterion Optim.* Cham, Switzerland: Springer, 2015, pp. 34–49, doi: 10.1007/978-3-319-15892-1_3.

[62] S. Wang, M. Gong, Y. Wu, and M. Zhang, "Multi-objective optimization for location-based and preferences-aware recommendation," *Inf. Sci.*, vol. 513, pp. 614–626, Mar. 2020.

[63] Y. Zhang, G.-G. Wang, K. Li, W.-C. Yeh, M. Jian, and J. Dong, "Enhancing MOEA/D with information feedback models for large-scale many-objective optimization," *Inf. Sci.*, vol. 522, pp. 1–16, Jun. 2020.

[64] J. Prieto and J. Gomez, "Hybrid adaptive evolutionary algorithm for multi-objective optimization," 2020, *arXiv:2004.13925*. [Online]. Available: http://arxiv.org/abs/2004.13925

[65] W.-Q. Feng and D.-W. Gong, "Multi-objective evolutionary optimization with objective space partition based on online perception of Pareto front," *Acta Automatica Sinica*, vol. 46, no. 8, pp. 1628–1643, 2020.

[66] Y. Yang, J. Liu, and S. Tan, "A multi-objective evolutionary algorithm for steady-state constrained multi-objective optimization problems," *Appl. Soft Comput.*, vol. 101, Mar. 2021, Art. no. 107042.

[67] R. Cheng, "Benchmark functions for the CEC 2017 competition on evolutionary many-objective optimization," School Comput. Sci., Univ. Birmingham, Birmingham, U.K., Tech. Rep. CSR-17-01, 2017.

[68] M. Abdel-Basset, R. Mohamed, M. Elhoseny, R. K. Chakrabortty, and M. Ryan, "A hybrid COVID-19 detection model using an improved marine predators algorithm and a ranking-based diversity reduction strategy," *IEEE Access*, vol. 8, pp. 79521–79540, 2020.

[69] M. Abdel-Basset, R. Mohamed, M. Elhoseny, A. K. Bashir, A. Jolfaei, and N. Kumar, "Energy-aware marine predators algorithm for task scheduling in IoT-based fog computing applications," *IEEE Trans. Ind. Informat.*, early access, Jun. 11, 2020, doi: 10.1109/TII.2020.3001067.

[70] A. Faramarzi, M. Heidarinejad, S. Mirjalili, and A. H. Gandomi, "Marine predators algorithm: A nature-inspired Metaheuristic," *Expert Syst. Appl.*, vol. 152, Aug. 2020, Art. no. 113377.
[71] M. Abdel-Basset, R. Mohamed, S. Mirjalili, R. K. Chakrabortty, and M. J. Ryan, "MOEO-EED: A multi-objective equilibrium optimizer with exploration–exploitation dominance strategy," *Knowl.-Based Syst.*, vol. 214, Feb. 2021, Art. no. 106717.
[72] W. L. Wang, W. K. Li, Z. Wang, and L. Li, "Opposition-based multi-objective whale optimization algorithm with global grid ranking," *Neurocomputing*, vol. 341, pp. 41–59, May 2019.
[73] M. Pescador-Rojas, "An overview of weighted and unconstrained scalarizing functions," in *Proc. Int. Conf. Evol. Multi-Criterion Optim.* Cham, Switzerland: Springer, 2017, pp. 499–513, doi: 10.1007/978-3-319-54157-0_34.
[74] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Comput. J.*, vol. 7, no. 4, pp. 308–313, 1965.
[75] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[76] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using Reference-Point-Based nondominated sorting approach—Part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014.
[77] A. J. Nebro, J. J. Durillo, J. Garcia-Nieto, C. A. C. Coello, F. Luna, and E. Alba, "SMPSO: A new PSO-based Metaheuristic for multi-objective optimization," in *Proc. IEEE Symp. Comput. Intell. Milti-Criteria Decision-Making*, Mar. 2009, pp. 66–73.
[78] S. Kukkonen and J. Lampinen, "GDE3: The third evolution step of generalized differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2005, pp. 443–450.
[79] X. Cai, Z. Mei, Z. Fan, and Q. Zhang, "A constrained decomposition approach with grids for evolutionary multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 4, pp. 564–577, Aug. 2018.
[80] J. J. Durillo and A. J. Nebro, "JMetal: A java framework for multi-objective optimization," *Adv. Eng. Softw.*, vol. 42, no. 10, pp. 760–771, Oct. 2011.
[81] N. Riquelme, C. Von Lucken, and B. Baran, "Performance metrics in multi-objective optimization," in *Proc. Latin Amer. Comput. Conf. (CLEI)*, Oct. 2015, pp. 1–11.
[82] W. Li, E. Ozcan, R. John, J. H. Drake, A. Neumann, and M. Wagner, "A modified indicator-based evolutionary algorithm (mIBEA)," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 1047–1054.
[83] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *Eur. J. Oper. Res.*, vol. 181, no. 3, pp. 1653–1669, Sep. 2007.
[84] M. S. Zapotecas and C. A. C. Coello, "A multi-objective particle swarm optimizer based on decomposition," in *Proc. 13th Annu. Conf. Genetic Evol. Comput.*, Jul. 2011, pp. 69–76.
[85] M. Aljanabi, M. A. Ismail, and V. Mezhuyev, "Improved TLBO-JAYA algorithm for subset feature selection and parameter optimisation in intrusion detection system," *Complexity*, vol. 2020, May 2020, Art. no. 5287684.

MOHAMED ABDEL-BASSET (Senior Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in operations research from the Faculty of Computers and Informatics, Zagazig University, Egypt. He is currently the Head of the Department of Computer Science, Faculty of Computers and Informatics, Zagazig University. He is working on the application of multi-objective and robust meta-heuristic optimization techniques. He has published more than 100 articles in international journals and conference proceedings. His current research interests include optimization, operations research, data mining, computational intelligence, applied statistics, decision support systems, robust optimization, engineering optimization, multi-objective optimization, swarm intelligence, evolutionary algorithms, and artificial neural networks. He is also an editor/a reviewer in different international journals and conferences.

REDA MOHAMED received the B.Sc. degree from the Department of Computer Science, Faculty of Computers and Informatics, Zagazig University, Egypt. His research interests include optimization, deep learning algorithms, swarm intelligence, evolutionary algorithms, and artificial neural networks.

SEYEDALI MIRJALILI (Senior Member, IEEE) received the B.Sc. degree in computer engineering (software) from Yazd University, the M.Sc. degree in computer science from Universiti Teknologi Malaysia (UTM), and the Ph.D. degree in computer science from Griffith University. He was a member of the Soft Computing Research Group (SCRG), UTM. He is currently a Lecturer with Griffith College, Griffith University. He is working on the application of multi-objective and robust meta-heuristic optimization techniques in computational intelligence. His research interests include robust optimization, multi-objective optimization, swarm intelligence, evolutionary algorithms, and artificial neural networks.

RIPON K. CHAKRABORTTY (Member, IEEE) received the B.Sc. and M.Sc. degrees in industrial and production engineering from the Bangladesh University of Engineering and Technology, in 2009 and 2013, respectively, and the Ph.D. degree in computer science from the University of New South Wales (UNSW Australia), Canberra, in 2017. He is currently a Lecturer of System Engineering and Project Management with the School of Engineering and Information Technology, UNSW Australia. He has written two book chapters and over 50 technical journal and conference papers. His research interests include operations research, optimization problems, project management, supply chain management, and information systems management.

MICHAEL RYAN (Senior Member, IEEE) received the bachelor's and master's degrees in engineering, and the Ph.D. degree in philosophy from UNSW Canberra. He did formal engineering management training, U.K., for two years. He is currently a Lecturer and a Regular Consultant in a range of subjects, including communications systems, systems engineering, requirements engineering, and project management. He is also the Director of the Capability Systems Centre, University of New South Wales, Canberra. He is a Fellow of Engineers Australia, the International Council on Systems Engineering, and the Institute of Managers and Leaders, and a Co-Chair of the Requirements Working Group, International Council on Systems Engineering (INCOSE).