Metaheuristic approaches for solving multiobjective optimization problems

Selim YILMAZ[1,2], Sevil SEN[1]

[1]WISE Lab., Department of Computer Engineering, Hacettepe University, Ankara, Turkey

[2]Department of Software Engineering, Muğla Sıtkı Koçman University, Muğla, Turkey

selimyilmaz@mu.edu.tr

ssen@cs.hacettepe.edu.tr

Abstract Multi-objective optimization problems (MOOPs) require to optimize two or more, often conflicting objectives. The wide application of MOOPs have attracted the researcher's attention from academy and industry; and therefore, a great deal of effort has been put to develop effective approaches towards the MOOPs. In this study, a new meta-heuristic approach is introduced: Multi-Objective Electric Fish Optimization (MOEFO). The proposed approach is based on the Electric Fish Optimization (EFO) algorithm, a recently proposed meta-heuristic algorithm for single-objective problems. Since EFO has achieved a significant performance on solving different types of problems such as constrained and unconstrained problems, it is extended here for solving MOOPs efficiently. The proposed approach is compared with well-known meta-heuristics in the literature, and the experimental results show that MOEFO is among the best algorithms on solving MOOPs within a competitive running time. Moreover, it becomes very competitive on solving challenging Many-objective Optimization Problems (MaOPs) having four or more objectives.

Keywords: Electric Fish Optimization algorithm, meta-heuristics, multi-objective optimization, many-objective optimization, Pareto-based algorithm.

1. Introduction

Many real-life problems have at least two objectives that stay in conflict with each other. For example, many solutions consider not only the accuracy of the proposal, but also its efficiency. Multi-objective optimization aims at optimizing these conflicting objectives simultaneously. To discover solutions for MOOPs so many meta-heuristic approaches have been proposed in the literature, and due to their powerful capability in dealing with this kind of problems, the new meta-heuristics are emerging in the recent years. These approaches are mainly classified into the following five classes: *aggregating*, *lexicographic*, *subpopulation*, *Pareto-based*, and *hybrid* approaches. Among them, the Pareto-based approaches are the most applied methods in the literature [1-3]. In Pareto-based approaches, the solution to MOOPs is generally not unique, but a set of optimal, non-dominated solutions called Pareto front. Hence, the Pareto front represents different trade-offs among conflicting objectives that decision-makers could choose the most appropriate solution according to their needs.

This paper presents a novel meta-heuristic algorithm (MOEFO), which is based on EFO inspired by electric fish in the nature [4]. Nocturnal electric fish have very poor eyesight and live in muddy, murky water, where the visual sense is very limited. Therefore, they rely on their species-specific ability called electrolocation to perceive their environment. The active and passive electrolocation capabilities of such fish are believed to be good candidates to perform local and global search, respectively. Hence it is proved in [4] that an algorithm based on such capabilities of electric fish is very successful on solving

complex problems. The main motivation of this study is to enhance this promising algorithm for solving MOOPs.

In EFO, every individual carries two information called amplitude and frequency, which represent the degree of proximity of the fish (the candidate individual) to the best prey source (the global optimum). The frequency plays a key role in EFO to balance exploitation and exploration, and is used to determine whether an individual will perform active or passive electrolocation. It enforces better individuals (active individuals), which are most likely to be in the vicinity of promising regions, to exploit their neighborhood, and it leads other individuals (passive individuals) to explore the search space so that they discover new regions. The amplitude, however, adjusts the length of search areas in which the active individuals perform local search whereas it determines the likelihood of active mode individuals to be sensed by passive mode individuals, which therefore promotes the better regions to be more exploited. That's because active mode individuals likely attract more individuals through their long-lasting higher amplitudes.

In order to adapt and improve EFO for solving MOOPs, some modifications have been made to the EFO algorithm. While the active and passive electrolocation phases of EFO are mainly kept the same in the proposed MOEFO algorithm, some adaptations are made in order to keep a set of non-dominated solutions instead of a single solution by using a dominance-based selection. Here, not only EFO is adapted for solving MOOPs but also the complexity of the algorithm is reduced for efficiency such as by adapting cellular-based approach [5, 6], which places individuals in a static toroidal mesh topology at the initialization. As a result of all these modifications, an effective and efficient multi-objective approach is introduced and shared with the community [7].

The proposed MOEFO is compared with well-known meta-heuristic approaches in the literature, namely Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [8], Strength Pareto Evolutionary Algorithm-II (SPEA2) [9], Indicator-based Evolutionary Algorithm (IBEA) [10], Multi-objective Evolutionary Algorithm based on Decomposition (MOEAD) [11], Multi-objective Cellular Genetic Algorithm (MOCell) [12], Generalized Differential Evolution 3 (GDE3) [13], and Optimized Multi-Objective Particle Swarm Optimizer (OMOPSO) [14]. The rationale behind the selection of these meta-heuristics as competitor algorithms is their popularity and high adoption in the studies carried out for performance assessment in the literature. Hypervolume (HV), SPREAD, EPSILON, Inverted Generational Distance (IGD) metrics are used in evaluations. The experimental results show that MOEFO outperforms its competitors in all metrics other than the SPREAD. It takes the third place on evaluations that are made by using the SPREAD metric. The results show that the solutions found by the proposed MOEFO algorithm are able to converge better to the Pareto-fronts and that they are much more diverse than those found by the competitor algorithms. Moreover, it has a comparable running time with the competitor algorithms.

The proposed MOEFO algorithm is also explored on MaOPs, which are recently welcomed in the literature as more challenging tasks. Unlike to the MOOPs having two to three objectives, MaOPs possess four or more objectives; hence they introduce some challenges such as diverseness and convergence for most of the existing meta-heuristics. The experimental results show that MOEFO algorithm is better than the competitors on some MaOPs and is very competitive on other MaOPs according to the IGD metric.

The rest of this paper is organized as follows: background information is given in the following subsection. The related studies in the literature are summarized in Section 2. An overview of EFO is

given in Section 3. The MOEFO algorithm is introduced in Section 4. Section 5 presents the experimental settings, then discusses experimental results with statistical findings. Finally, this study is concluded in Section 6.

1.1. Definitions

In this section, some definitions used throughout the paper are presented. These definitions are the basic concepts of Pareto-based multi-objective optimization approaches. A general MOOP is expressed in the following form:

Definition 1 Multi-Objective Optimization Problem: A type of problem where the objective is to find a solution $x^* = [x_1^*, x_2^*, \dots, x_k^*]$ that minimizes a number of functions $f(x^*)$ satisfying the p inequality constraints $g(x^*)$ and q equality constraints $h(x^*)$. Considering x is a decision variable set, a function and constraint vectors can be written as follows (in case when the goal is maximization, *minimize* should be changed to *maximize*):

$$\begin{aligned} minimize\ f(x) &= [f_1(x), f_2(x), \dots, f_n(x)], \\ subject\ to\ g(\mathrm{x}) &= \left[g_1(x), g_2(x), \dots, g_p(x)\right] \leq 0 \\ h(x) &= \left[h_1(x), h_2(x), \dots, h_q(x)\right] = 0 \end{aligned}$$

The region constrained by g and h is *feasible region* (Ω) and any solution (x) within this region (i.e., $x \in \Omega$) is called as *feasible solution*. The following Pareto definitions are made with respect to the notations in MOOP definition.

Definition 2 Pareto Dominance: A k-dimensional solution $u = [u_1, \dots, u_k]$ is said to dominate another solution $v = [v_1, \dots, v_k]$ (denoted as $u \prec v$); if $f(u)$ is partially less than $f(v)$, i.e., $\forall i$: $f_i(u) \leq f_i(v) \ \wedge \ \exists i$: $f_i(u) < f_i(v) \mid i \in \{1, \dots, n\};\ u, v \in \Omega$.

Definition 3 Pareto Optimal: A solution u is Pareto optimal if there is no other solution in Ω that dominates u, i.e., $\neg\exists \hat{u} \in \Omega : \hat{u} \prec u$.

Definition 4 Pareto Optimal Set: A set $\mathcal{P}^*$ that contains all the Pareto optimal solutions, i.e., $\mathcal{P}^* = \{u \in \Omega \mid \neg\exists \hat{u} \in \Omega : \hat{u} \prec u\}$.

Definition 5 Pareto Front: A set $\mathcal{PF}^*$ that contains all the objectives of Pareto optimal set ($\mathcal{P}^*$), i.e., $\mathcal{PF}^* = \{f(u) \mid u \in \mathcal{P}^*\}$.

2. Related Works

The complex nature of search space of some objectives in MOOPs such as non-differentiability, discontinuity makes traditional gradient-based techniques impossible or at least ineffective to apply. That's why, researchers have proposed many population-based heuristic optimization methods, which are not dependent on the characteristics of the search space. Such methods target to find as many Pareto-optimal solutions as possible. This paper summarizes the well-known studies in this research area.

The first use of heuristic algorithms for the solution of MOOPs dates back to the introduction of the Vector Evaluated Genetic Algorithm (VEGA) [15] to the literature. VEGA decomposes a problem with n objectives into n subproblems and splits population into n sub-populations. Hence, each decomposed subproblem is addressed only by the corresponding sub-population. The most drawback of VEGA is that it leads the entire population to converge to only one optimum due to using a selection

mechanism that often relies on best solutions. The lack of proper selection scheme in VEGA led the emergence of the Pareto concept [16]. MOEAD [11] is another method that decomposes the problem into smaller sub-problems. The main goal is to minimize the maximum margin between each objective and its corresponding reference value. Similar to VEGA, it decomposes the problem into several sub-problems and handles each of them simultaneously. The main drawback, however, is that its performance is directly proportional to the decomposition method employed.

Contrary to the approaches above, IBEA [10] handles a MOOP in a single population. In the calculation of fitness value, it does not consider the objectives directly, but use a particular performance metric obtained from the objectives such as HV, EPSILON, and aims to increase the quality of solutions with respect to the chosen metric. While parent and offspring individuals are constructing the next population, the poor solutions with regard to the chosen metric are eliminated to maintain the size of the population. However, the use of a single metric could lead to a poor performance with respect to other metrics. Similar to IBEA, Pareto-based SPEA2 [9], an efficiency-aware extension of SPEA [17], does not make directly use of the approximated Pareto front set for fitness evaluation either, but uses a fitness value calculated based on the Pareto domination count of a solution and its distance to other solutions. One of the main concerns about SPEA2 is that these two criteria used in fitness evaluation are treated evenly, which enables solutions that are completely dominated but farthest away from others to be kept in the population. NSGA-II [8] is the most popular Pareto-based algorithm in the literature. It is based on the Pareto dominance and distances of solutions in the objective space. Depending on the Pareto domination, NSGA-II splits the population into several fronts in which distance, called crowding distance, between the solutions is calculated. Different to SPEA2, non-dominated solutions that are in better fronts are allowed in NSGA-II to survive. If the size of next front exceeds the size of the population, the distance plays a key role in determining individuals to remain in the population. NSGA-III [18], is an improved version of NSGA-II, which is proposed to primarily solve MaOPs. Even if NSGA-III follows the same search framework as that in NSGA-II, it slightly differs in the selection phase. Unlike to the crowding distance, NSGA-III relies on a set of well-spread reference points to keep the population as diverse as possible throughout the optimization. The multi-objective GDE3 [13] algorithm is a developed version of the multi-objective GDE [19] algorithm. GDE algorithm replaces only the fitness-based greedy selection scheme with the Pareto dominance-based selection. GDE3 is a very similar algorithm to NSGA-II. It employs Pareto front and crowding distance-based survival schemes to keep a more diverse population in the objective space. MOCell [12] is also similar to NSGA-II in the selection of solutions to survive for the next generations. However, it applies the basic generic operators of GA in order to breed offspring in only close neighborhood. This neighborhood is based on a pre-defined grid topology.

In addition to these well-known evolutionary-based algorithms, swarm-based heuristics have also been proposed to solve MOOPs. Such heuristics are generally proposed for solving single-objective problems, then their extensions are introduced for MOOPs. Particle Swarm Optimization (PSO) [20], Artificial Bee Colony (ABC) [21] are the most popular swarm-based algorithms in the literature. Therefore, they have mostly been adapted for solving MOOPs. The initial attempt to adapt PSO for solving MOOPs is MOPSO [22] algorithm. MOPSO is proposed as a Pareto-based approach and it makes a slight modification on the search operators in the single-objective PSO algorithm. With this modification, the leader archive in MOPSO is determined according to the Pareto dominance. Another most popular PSO-based multi-objective algorithm is OMOPSO [14]. OMOPSO, similar to MOPSO, considers the Pareto dominance of solutions for the construction of leader archive. Differently from

MOPSO, crowding distance plays a role in OMOPSO when non-dominated solutions exceed the pre-defined archive size. Decomposition-based PSO (MPSO/D) [23], Speed-constrained PSO (SMPSO) [24], Vector Evaluated PSO (VEPSO) [25], Dynamic Neighborhood PSO (DN-PSO) [26] are among other popular PSO variants proposed for MOOPs.

The first modification to ABC for solving MOOPs is introduced as Vector Evaluated ABC (VEABC) [27], which is inspired by the VEGA and VEPSO algorithms. Therefore, similar to these algorithms, VEABC splits the whole population into sub-populations which are responsible for handling different objectives of the problem. Multi-objective ABC (MOABC) [2] is another ABC-based multi-objective optimization algorithm. The search framework of the single-objective ABC is generally preserved in MOABC except the fitness proportionate selection scheme. It is replaced with a Pareto-dominance-based selection approach in MOABC. Asynchronous/synchronous (A/S), Pareto dominance/non-dominated sorting (PD/NS), MOABC algorithms (A-MOABC/PD, A-MOABC/NS, and S-MOABC/NS) are proposed as three different implementations of the single-objective ABC to extend the standard algorithm to handle MOOPs [28]. As in MOABC, the search framework employed in all variants is the same as the standard ABC algorithm. However, they show difference in their selection schemes both from each other and from the standard ABC algorithm. Non-dominated Sorting-based ABC (NSABC) [29], Division-based multi-objective ABC (dMOABC) [30], Elitism-based multi-objective ABC (eMOABC) [31] are other popular ABC-based multi-objective algorithms.

As it is pointed out earlier, PSO and ABC are the, maybe the most, popular algorithms that have often been used to solve MOOPs in the literature mostly due to their easy-to-implement structures and satisfying performances on different types of problems. Refer, respectively, to [1, 32] for a detailed review of PSO- and ABC-based multi-objective optimization algorithms proposed in the literature. Apart from them, there are also multi-objective optimization algorithms that are built on other swarm-based single-objective optimization algorithms. These include Multi-objective Gravitational Search Algorithm (MOGSA) [33], Multi-objective Grey Wolf Optimization (MOGWO) [34], Multi-objective Firefly Algorithm (MOFA) [35], Multi-objective Teaching Learning-based Optimization (MOTLBO) [36].

3. An Overview of Electric Fish Optimization

EFO algorithm is based on the following characteristics of electric fish: *i*) active electrolocation, *ii*) passive electrolocation, which is dependent on the activity of electric organ discharge (EOD), *iii*) EOD frequency, and *iv*) EOD amplitude behaviors. The local and global search is ensured in EFO through the modelled active and passive electrolocation, respectively. The balance between local and global search, however, is determined by EOD frequency. Because they are expected to further exploit the promising area in the vicinity of them, individuals with a higher frequency perform local search; while the other individuals perform global search. EOD amplitude is used in order to determine effective range in local search and the probability of neighboring individuals' to be sensed in global search.

3.1. Initialization of Population

EFO algorithm starts by generating a population of individuals (N) randomly, which is common in most heuristic algorithms:

$$x_{ij} = x_{\min j} + \phi\left(x_{\max j} - x_{\min j}\right) \tag{1}$$

where x_{ij} represents the position of the ith individual in the population of size $|N|$ ($i = 1, 2, \dots, |N|$) in the d-dimensional search space. $x_{\min j}$ and $x_{\max j}$ are the lower and upper boundaries for dimension

$j \mid j \in 1, 2, \ldots, d$, respectively. $\phi \in [0, 1]$ is a random value drawn from a uniform distribution. The algorithm then calculates the frequency (f) and amplitude (A) values of every individual with regard to Equation 2 and 3, respectively.

$$f_i^t = f_{\min} + \left(\frac{fit_{worst}^t - fit_i^t}{fit_{worst}^t - fit_{best}^t} \right) (f_{\max} - f_{\min}) \tag{2}$$

$$A_i^t = \alpha A_i^{t-1} + (1 - \alpha) f_i^t \tag{3}$$

where fit_{worst}^t and fit_{best}^t are, respectively, the worst and best fitness values, whereas fit_i^t is the fitness value of the ith individual at iteration t. $\alpha \mid \alpha \in [0, 1]$ is a constant value that determines the magnitude of the previous amplitude value. The initial amplitude value of the ith individual is set to its own initial frequency value f_i.

3.2. Active and Passive Electrolocation Phases

After the initialization of individuals, the population is divided into two groups based on the frequency values of individuals in the population: active and passive. The individuals in these groups perform either active or passive electrolocation according to the group they are assigned to. Hence, the search continues in parallel manner through passive (N_P) and active individuals (N_A), and creates a new population by updating individuals' frequency and amplitude values. These steps are iterated for each new population until the termination criterion is met.

Active individuals are allowed to search only their very vicinity as in the nature. Hence, they fulfill the local search ability of EFO algorithm. An individual in active mode first determines its active sensing range (r_i) depending on its amplitude value (Equation 4), examines its neighborhood, and then evolves a new candidate solution through randomly selecting one of its neighbors (k) (Equation 5), or through a random walk in case where no neighbor exists in its vicinity (Equation 6).

$$r_i = \left(x_{\max j} - x_{\min j} \right) A_i \tag{4}$$

$$x_{ij}^{cand} = x_{ij} + \varphi \left(x_{kj} - x_{ij} \right) \tag{5}$$

$$x_{ij}^{cand} = x_{ij} + \varphi r_i \tag{6}$$

where $\varphi \in [-1, 1]$ in Equation 5 and 6 is a random number generated from a uniform distribution and x_{ij}^{cand} represents newly evolved candidate solution.

Passive individuals, however, can easily exceed their vicinity as opposed to active individuals. They fulfill the global search ability of EFO algorithm. They perceive their conspecific active individuals with a probability, and then change their locations. The being perceived probability of active individuals is directly proportional to their own amplitude value and their distance to the target passive individual as given in Equation 7. Here, EFO algorithm applies Euclidean distance formula in order to calculate the distance between two individuals in the space.

$$p_k = \frac{A_k / d_{ik}}{\sum_{j \in N_A} A_j / d_{ij}} \tag{7}$$

A reference location vector x_r is then built from perceived K individuals (Equation 8) then a new auxiliary solution (x^{new}) is generated from this reference vector (Equation 9).

$$x_{rj} = \frac{\sum_{k=1}^{K} A_k x_{kj}}{\sum_{k=1}^{K} A_k} \tag{8}$$

$$x_{ij}^{new} = x_{ij} + \varphi\left(x_{rj} - x_{ij}\right) \tag{9}$$

EFO employs an acceptance condition (Equation 10) on the generated solution in order to avoid a case in which a passive mode individual with higher frequency loses its promising location information completely to happen.

$$x_{ij}^{cand} = \begin{cases} x_{ij}^{new} & rand_j(0,1) > f_i \\ x_{ij} & otherwise \end{cases} \tag{10}$$

Lastly, in order to further increase the diversity in a population, EFO employs the following equation that stochastically enables one parameter (j) of newly evolved solution to be modified:

$$x_{ij}^{cand} = x_{\min j} + \phi\left(x_{\max j} - x_{\min j}\right) \tag{11}$$

EFO has only two parameters (α and K) that need to be set. However, EFO is shown not to be very sensitive to changes in parameters [4]. It is also stated that distance calculation between individuals brings about an increase in the complexity of the algorithm; but leads EFO to show superior performance, particularly on complex problem. The effective performance of EFO on unconstrained and constrained single-objective problems is statistically revealed in detail in [4]. Refer [4] for pseudo-code and further information on EFO.

4. Multi-objective Electric Fish Optimization Algorithm

In this study, EFO is extended for solving MOOPs. While the general framework based on local and global search is kept similar to as in EFO, some modifications have been made to the base algorithm in order to adapt it for MOOPs (1-2). Furthermore, by taking into account the more complex search space of such problems, some changes are applied in order to strengthen the exploration and exploitation search capabilities of the algorithm (3-4), and to reduce the complexity of the algorithm (5). The modifications to the EFO algorithm are summarized in the followings:

i. The frequency updating of individuals (Equation 2) has been modified. Hence the single fitness proportionate-based approach is replaced with a ranking- and crowding distance-based approach.
ii. The greedy selection has been replaced with a dominance-based selection and an external archive for storing Pareto optimal solutions is introduced.
iii. The conditional acceptance in passive search has been improved in order to support promising solutions to perform their search in the vicinity of a non-dominated solution.

iv. Binary tournament selection has been employed in active and passive search to ensure a fine-tuned local and global search.

v. The distance calculation in EFO has been excluded here for the sake of efficiency of the proposed algorithm. Instead, a *cellular-based* approach has been adopted.

4.1. Population Initialization

Here, the individuals are randomly generated as in EFO. So, Equation 1 has been applied for the formation of initial population. In EFO, an individual needs to measure its distance to the rest of the population for finding neighboring individuals in the sensing/active range. However, this distance calculation at each iteration increases the computation complexity of the algorithm, which is also stated in [4]. Therefore, it is excluded in MOEFO; instead the cellular-based approach is employed. Neighborhood in this approach has been organized in a random manner and so every individual has been directly connected with eights neighbors in this phase.

In the area of optimization, cellular-based approach was emerged from parallelization of genetic algorithm [5, 6]. This approach relies on the placement of every node, which corresponds to individuals, in a toroidal mesh topology and edges directly connected to nodes represent the one-hop neighbors of individuals.

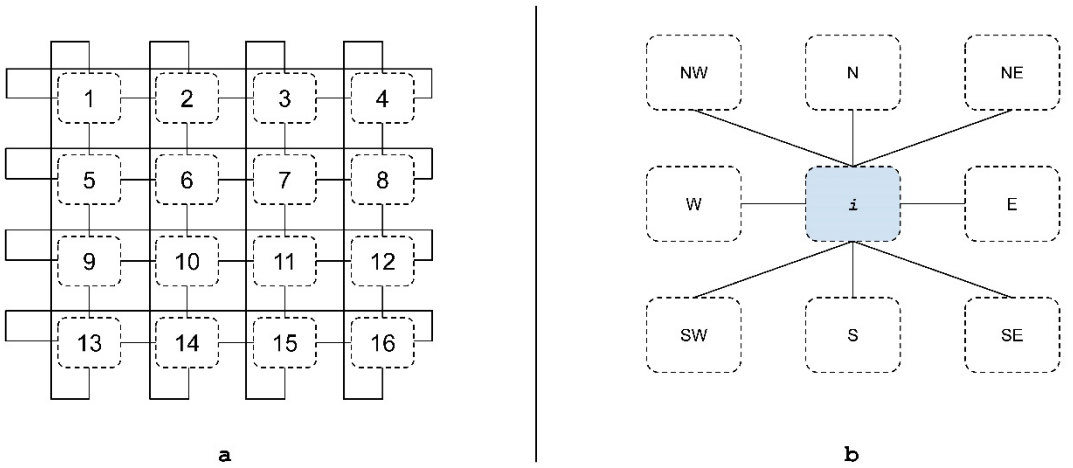

Figure1. a) Toroidal mesh topology used in cGA. b) A neighborhood demonstration of *i*th individual. (N: North, S: South, W: West, E: East)

Figure 1-a shows an exemplar mesh topology with 16 nodes. As a result of this topology, every individual intensively uses its nearby neighbors. In this study, a topology with eight neighbors are employed as shown in Figure 1-b.

Even if the neighborhood in this approach has no relation to the geographical position of neighbor individuals, it still well balances the *exploration* and *exploitation*. While exploration is ensured by a slow diffusion of solution through the population; exploitation is ensured by application of search operators within the neighborhood [12]. It is worth stating here that not only distance calculation in EFO establishes the neighborhood, but also maintains the 'explore first exploit later' approach [4]. The cellular-based strategy employed here also contributes the 'explore first exploit later' approach as the neighborhood of every individual has initially been organized in a random manner. This allows every

individual to communicate with the randomly chosen individuals at the beginning of the iterations and to exploit them towards the end of the iterations. In addition to the cellular-based approach, an archive has been created in this phase in order to store Pareto-optimal solutions.

4.2. Frequency Update

A substantial change has been applied to the frequency updating mechanism as only single objective was used in EFO for the calculation of individuals' frequencies. This approach is not applicable in MOOPs where at least two conflicting objectives take part. That's why, a modification based on Pareto ranking and crowding distance of every individual is proposed. While f_{min} and f_{max} represent the worst and the best respectively in EFO, they are evenly split into intervals that represent the total Pareto ranking of the population here. Individuals in the first ranking are placed into the first interval, those in the second ranking are placed in the second interval, and so on. Then crowding distance is applied separately on every sub-populations in the intervals such that the higher distance an individual has the closer to the upper limit of that interval it has been placed. The calculation of the frequency of ith individual (f_i) in the population is given in Equation 12.

$$f_i = \left(1 - \left(\frac{front_i - 1}{front_{max}}\right) - dist_i\right) \tag{12}$$

where $front_i$ and $front_{max}$ represent the front at which the ith individual belongs to and the total number of fronts, respectively; while $dist_i$ is a crowding distance measurement for the ith individual ($CrowDist_i$) normalized for $front_i$. The calculation of $dist_i$ is given as follows:

$$dist_i = \left(\left(1 - \frac{CrowDist_i - CrowDist_{min}}{CrowDist_{max} - CrowDist_{min}}\right) / front_{max}\right) \tag{13}$$

where $CrowDist_{min}$, and $CrowDist_{max}$ represents, respectively, the value of minimum and maximum crowding distance measurements obtained from $front_i$. Figure 2 demonstrates the frequency update mechanism in MOEFO. The figure is based on a scenario where the population is represented with four Pareto fronts in total such that each front comprises of different number of solutions. The proposed frequency update approach assigns the maximum and minimum frequency values to every front. Hence, individuals in the first front have a frequency value between 1.00 and 0.75, individuals in the second front have a frequency value between 0.75 and 0.5, and so on.

The distance of every individual is then calculated separately on every front with respect to the crowding distance measurement (refer to Equation 13) and then normalized. Here, the individuals having *Infinity* crowding distances represent the solutions where their objective vectors are located in the extreme points of the front, which are known to be solutions that increase the diversity of the population. In MOEFO, these individuals are excluded in distance calculation (Equation 13) and they are assigned to the maximum frequency value of the front where they belong to. Hence, these individuals are promoted to perform local search and their probability of being selected by other individuals are increased.

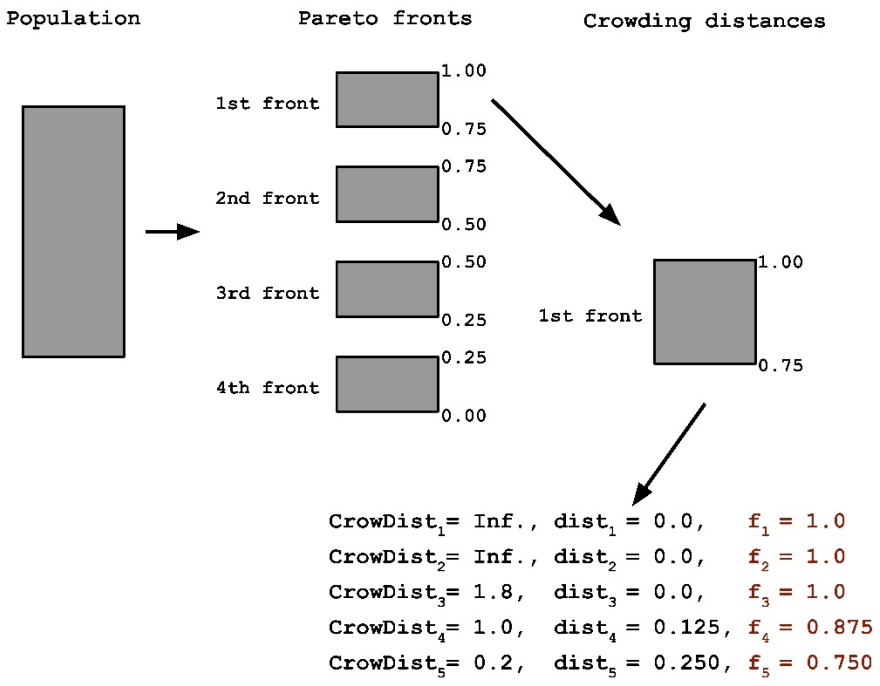

Figure 2. A demonstration of the frequency calculation strategy in MOEFO.

To sum up, as in EFO, this updating mechanism ensures better individuals to have higher frequency and thus enables them to perform local search, and vice versa. The update procedure for individuals' amplitudes remains same as in EFO, and hence it is calculated by using Equation 3 and 12.

4.3. Active and Passive Electrolocation Phases

Only few modification has been made on the search performed by active and passive individuals, which is mostly limited to the selection scheme employed in active and passive electrolocation phases.

The main role of active individuals in EFO is to perform local search since they take into account only the individuals in their very close neighborhood during the search. However, depending on the existence of neighbors that are evaluated by using Euclidean distance formula, individuals could perform random walk or neighbor exploitation that could occasionally lead the algorithm to contribute to global and local search during the initial and final iterations, respectively. However, MOEFO does not rely on Euclidean distance between individuals, but instead it makes use of one-hop neighbors that are assigned at the initialization (see Section 4.1). The proposed scheme contributes also to global search at the initial iterations, because the neighborhood is randomly established. Then, it contributes to local search as the neighbors continuously converge to the desired global minima. In order to further ensure a well-balanced search, *binary tournament selection* procedure has been used in neighbor selection because it first picks out two candidate solutions in a random manner (contributes to exploration) then returns the better one among them (contributes to exploitation). In MOEFO, crowding distance operator has been used as comparator. This operator considers only the front and the distance of the solutions. It prefers the ith solution, if it belongs to lower front than the jth solution (i.e., $front_i < front_j$). If both solutions belong to the same front, it chooses the solution with lesser crowded region (i.e., $front_i = front_j \wedge CrowDist_i > CrowDist_j$). If both solutions belong to the same front and have the same distances, it returns a random solution (see Equation 14). After this

selection mechanism, ith active individual applies the same formula as in EFO (see Equation 5) to evolve a new candidate solution.

$$return \begin{cases} i & \left(front_i < front_j\right) \vee \left(\left(front_i = front_j\right) \wedge \left(CrowDist_i > CrowDist_j\right)\right) \\ j & \left(front_j < front_i\right) \vee \left(\left(front_i = front_j\right) \wedge \left(CrowDist_j > CrowDist_i\right)\right) \\ i \; or \; j & otherwise \end{cases} \quad (14)$$

In passive search, EFO takes all the active individuals in the population into account in order to perform global search. This mechanism is also adopted in MOEFO. However, the selection approach applied to passive individuals has been modified, since distance-based search is excluded in MOEFO. As in EFO, K active individuals are probabilistically chosen from the population in MOEFO by applying K times *binary tournament selection*. The selection procedure here is based on the amplitude values that are implicitly determined by the crowding distance measurement (see Section 4.2). The reason behind using amplitudes in the selection instead of crowding distance metric as in active search is that amplitude values of individuals do not change instantaneously. This adoption enables currently poor but formerly promising individuals being selected, which further promotes exploration capability of MOEFO.

The other modification has been made on the conditional acceptance condition given in Equation 10 by replacing it with Equation 15. This modification simply states that the probability of a promising solution with higher frequency being evolved through its neighbors is lowered but that through a Pareto-optimal solution (a) from *archive* is considerably increased.

$$x_{ij}^{cand} = \begin{cases} x_{ij}^{new} & rand_j(0,1) > f_i \\ x_{ij} + \varphi(x_{aj} - x_{ij}) & otherwise \end{cases} \quad (15)$$

The final step in one iteration is on the replacement strategy of existing solutions with evolved candidate solutions. In EFO, *greedy selection* relying on only a single objective has been used. However, it is not suitable for MOOPs

Therefore, a dominance-based comparator, which prefers a non-dominated solution over a dominated solution, is used in this study. So, a candidate solution can be replaced with its parent solution only when it dominates, otherwise it can be replaced with its worst neighbor or it is discarded when it is worse than its worst neighbor from the point of domination view. The pseudocode of the proposed MOEFO algorithm is provided in Figure 3.

Algorithm: Pseudocode of MOEFO algorithm

```
Input: population size |N|, objective size |M|
Output: archive
population ← Initialize population;
Update frequency;
archive ← Update archive;
repeat
    for i = 1 to |N| do
        individual ← population[i];
        offspring ← {};
        if individual.frequency > rand then
            offspring ← active search(individual);
        else
            offspring ← passive search(individual);
        end
        if offspring ≺ individual then
            individual ← offspring;
        else if offspring ≺ individual.neighbor[worst] then
            individual.neighbor[worst] ← offspring;
        end
    end
    Update frequency;
    archive ← Update archive;
until termination criterion is met;
return archive;
```

Figure 3. Pseudocode of MOEFO algorithm.

5. Experiments

The performance of the proposed MOEFO algorithm on MOOPs and MaOPs has been evaluated by some metrics that are intrinsic for such problem types. The following sections introduce benchmark problems, performance metrics, algorithms used for comparison and their settings, respectively. Finally, the experimental results with a discussion are presented.

5.1. Benchmark Problems

In this study, we have used 64 well-known unconstrained MOOPs having different characteristics. Among these, 30 problems are taken from DTLZ(Deb-Thiele-Laumanns-Zitzler) [37], LZ09 (Li-Zhang) [38], WFG (Walking FishGroup) [39], and ZDT (Zitzler-Deb-Thieler) [40], 10 complex MOOPs are taken from the CEC09 algorithm contest [41], and 24 multi-modal MOOP functions (MMF), which are proposed for the CEC2020 contest, are taken from the technical report given in [42]. MMFs differ from MOOPs in that there exists at least one local Pareto optimal solution, not dominated by any solution in the neighborhood, and at least two global optimal solutions, not dominated by any solution in the search space, corresponding to the same Pareto front in the objective space. The majority of these problems have become the standard problems used in the literature to conduct a fair comparison. They differ from each other in search space, number of parameters to optimize (6 through 30), and number of objectives (2 and 3). Table 1 gives the basic characteristics of these problems. It is worth stressing on this table that the equations of functions f, f_l, or f_a in MMF family are same but their reference data are different. For example; MMM15, MMM15_l, and MMF15_a have same equation to be optimized; however, they differ only in their reference data.

As for the performance evaluation on MaOPs, 10 different benchmark problems with different challenging features, which are proposed in [43], are used. Table 2 gives specific challenges of these problems. Refer to [43] for mathematical formulations and detailed discussions of these problems.

5.2. Performance Metrics

The performance metrics applied on MOOPs differ from the metrics used for evaluating single-objective problems. Without the loss of generality, an algorithm is regarded as the best performing algorithm on a single-objective problem when its final solution is closest to the global optimum in comparison to others. However, for MOOPs, it depends on how well $\mathcal{PF}^*$ computed by non-dominated solutions of an algorithm approximates to the true Pareto front ($\mathcal{TPF}^*$) of the problem. The following metrics reveal the convergence performance of an algorithm on MOOPs:

Table 1. The characteristics of MOOPs used in the experiments.

Family	Problem	O	V	Range
DTLZ	1	3	7	$0 \leq x_1, x_2, \dots, x_V \leq 1$
	2-6		12	
	7		22	
LZ09	1,7,8	2	10	$0 \leq x_1, x_2, \dots, x_V \leq 1$
	2-5,9	2	30	
	6	3	10	
WFG	1-9	2	6	$0 \leq x_k \leq 2 \times k, k \in \{1,2, \dots, V\}$
ZDT	1-3	2	30	$0 \leq x_1, x_2, \dots, x_V \leq 1$
	4		10	$0 \leq x_1 \leq 1, -5 \leq x_2, \dots, x_V \leq 5$
	6		10	$0 \leq x_1, x_2, \dots, x_V \leq 1$
CEC09	1,2,5-7	2	30	$0 \leq x_1 \leq 1, -1 \leq x_2, \dots, x_V \leq 1$
	3	2		$0 \leq x_1, x_2, \dots, x_V \leq 1$
	4	2		$0 \leq x_1 \leq 1, -2 \leq x_2, \dots, x_V \leq 2$
	8-10	3		$0 \leq x_1, x_2 \leq 1, -1 \leq x_3, \dots, x_V \leq 1$
MMF	1	2	2	$1 \leq x_1 \leq 3, -1 \leq x_2 \leq 1$
	1_e	2	2	$1 \leq x_1 \leq 3, e^{-3} \leq x_2 \leq e^3$
	2	2	2	$0 \leq x_1 \leq 1, 0 \leq x_2 \leq 2$
	4	2	2	$-1 \leq x_1 \leq 1, 0 \leq x_2 \leq 2$
	5	2	2	$-1 \leq x_1 \leq 3, 1 \leq x_2 \leq 3$
	7	2	2	$1 \leq x_1 \leq 3, -1 \leq x_2 \leq 1$
	8	2	2	$-\pi \leq x_1 \leq \pi, 0 \leq x_2 \leq 9$
	10,10_l	2	2	$0.1 \leq x_1, x_2 \leq 1.1$
	11,11_l	2	2	$0.1 \leq x_1, x_2 \leq 1.1$
	12,12_l	2	2	$0 \leq x_1, x_2 \leq 1$
	13,13_l	2	3	$0 \leq x_1, x_2, x_3 \leq 1$
	14,14_a	3	3	$0 \leq x_1, x_2, x_3 \leq 1$
	15,15_a,15_l,15_a_l	3	3	$0 \leq x_1, x_2, x_3 \leq 1$
	16_l1,16_l2,16_l3	3	3	$0 \leq x_1, x_2, x_3 \leq 1$

O: Number of objectives, V: Number of Variables.

5.2.1. Hypervolume (HV):

It measures the union of the volume between every member in $\mathcal{PF}^*$ and a reference point r which represents a vector comprising the worst values of each objective. Therefore, the higher HV implies that $\mathcal{PF}^*$ well represents $\mathcal{TPF}^*$. HV is calculated by the following equation:

$$HV = volume\left(\bigcup_{i=1}^{|\mathcal{PF}^*|} v_i\right) \tag{16}$$

Table 2. The characteristics of MaOPs used in the experiments [21].

Problem	Characteristics	Range
MaOP1	Inverse of simplex, objective scales, multimodality	$0 \leq x_1, x_2, \ldots, x_V \leq 1$
MaOP2	Complicated PS	
MaOP3	Complicated PS, bias	
MaOP4	Complicated PS, bias	
MaOP5	Complicated PS, degeneracy	
MaOP6	Complicated PS, degeneracy	
MaOP7	Complicated PS, local degeneracy	
MaOP8	Complicated PS, local degeneracy	
MaOP9	Complicated PS, local degeneracy	
MaOP10	Complicated PS, local degeneracy	

5.2.2. SPREAD

It reveals the degree of spread of $\mathcal{PF}^*$ throughout the objective space through the following equation:

$$\Delta = \frac{\sum_{i=1}^{m} d(e_i, \mathcal{PF}^*) + \sum_{x \in \mathcal{PF}^*} \left| d(e_i, \mathcal{PF}^*) - \hat{d} \right|}{\sum_{i=1}^{m} d(e_i, \mathcal{PF}^*) + |\mathcal{PF}^*| \hat{d}} \tag{17}$$

where e_i stands for the ith extreme point in the mth objective in $\mathcal{TPF}^*$. $d(e_i, \mathcal{PF}^*)$ refers to the minimum distance between the extreme points in the set $\mathcal{PF}^*$ and $\mathcal{TPF}^*$, whereas $\hat{d}$ is the mean of the distances between every point in $\mathcal{PF}^*$ and in $\mathcal{TPF}^*$.

5.2.3. EPSILON

It calculates the minimum factor (ϵ) that is necessary for at least one solution in $\mathcal{PF}^*$ to dominate all the vectors in $\mathcal{TPF}^*$. EPSILON is calculated according to the given equation:

$$E(\mathcal{PF}^*, \mathcal{TPF}^*) = \inf_{\epsilon \in \mathbb{R}^+} \{\forall \mathbf{p} \in \mathcal{TPF}^*, \exists \mathbf{v} \in \mathcal{PF}^* : \mathbf{v} \prec_\epsilon \mathbf{p}\} \tag{18}$$

here m refers to the number of objective functions, whereas ϵ represents a very small value used for a tolerance.

5.2.4. Inverted Generational Distance (IGD)

IGD is a good indicator that reveals how diverse the solutions in objective space is and how well the solutions can converge. The small value of IGD suggests that the solution set obtained from the objective space is very close to $\mathcal{TPF}^*$. IGD is calculated as follows:

$$IGD(\mathcal{PF}^*, \mathcal{TPF}^*) = \frac{\sum_{v \in \mathcal{TPF}^*} d(v, \mathcal{PF}^*)}{|\mathcal{TPF}^*|} \quad (19)$$

here $d(v, \mathcal{PF}^*)$ gives the minimum Euclidean distance between the solution v and every point from the set $\mathcal{PF}^*$.

5.3. Competitor Algorithms

For a comprehensive performance evaluation, well-known evolutionary- and swarm-based multi-objective optimization algorithms have been used as competitor methods against MOEFO. These include NSGA-II [8], NSGA-III [18], SPEA2 [9], IBEA [10], MOEAD [11], MOCell [12], GDE3 [13], and OMOPSO [14]. The majority of the competitor algorithms are Pareto-based relying mostly on Pareto-domination and crowding distance measurements for selection of solutions for the next generations. Among them, IBEA and MOEAD are not Pareto-based which, respectively, use a fitness-based greedy comparison to select the individuals for the survival. While fitness calculation in IBEA depends on the indicator metric value (such as HV) of the individuals, it is maximum difference between individual's jth objective and ideal point for jth objective found by the population. Typical crossover and mutation operators are employed for breeding new individuals in these algorithms. Some of these algorithms (i.e., MOEAD, MOCell, and OMOPSO) possess an external memory (known as archive) to keep non-dominated solutions until the end of the search, whereas the remaining algorithms output only the non-nominated solutions found at the end of the search. Among these competitor approaches, MOCell and MOEAD make use of a neighboring structure and every individual in these algorithms performs the search in its neighborhood.

Table 3. The comparative features of competitor algorithms.

Algorithm	**Category**	**Selection** Pareto-domination (●) Greedy comparison (●) Crowding distance (○)	**Search Procedure** Crossover (■) Mutation (□)	**Archive**
NSGAII	Pareto-based	● ○	■ □	
NSGAIII	Pareto-based	● ○	■ □	
SPEA2	Pareto-based	●	■ □	
IBEA	Indicator-based	●	■ □	
MOEAD	Decomposition-based	●	■ □	✓
MOCell	Pareto-based	● ○	■ □	✓
GDE3	Pareto-based	● ○	■	
OMOPSO	Pareto-based	● ○	* □	✓

*: Crowding distance-based search procedure of the standard PSO [15].

These competitor algorithms are reviewed in Section 2 and their features are outlined in comparative manner in Table 3. jMetal (Meta-heuristic Algorithms in Java) [44, 45], a Java-based framework for multi-objective optimization with meta-heuristics, is used for the implementation of these competitor algorithms. jMetal provides some facilities such as state-of-the algorithms, most-popular benchmark problems, and the like. Refer to [44] for detailed description of this framework.

The detailed experimental settings and performance comparisons are given below separately for MOOP and MaOP benchmark sets.

5.4. Performance Evaluation on MOOP Benchmark Sets

5.4.1. Parameter tuning

The parameter values of the competitor algorithms are taken from the default values of jMetal. As for the parameter set of the proposed MOEFO algorithm (i.e., α and K), the design of experiment (DoE) methodology has been conducted in order to find out their optimal values. DoE methodology first uses a second order linear model in order to learn the relation between input (parameter values) and the output of the algorithm through a coefficient vector. It then uses quadratic programming with that coefficient vector to find the approximation to the optimal parameter.

As it would be too exhaustive to run DoE with every possible parameter setting, three-level full factorial design, in which each parameter is set to three levels (referred to as high, intermediate, and low), has been adopted here. To construct input and output data, MOEFO has been run 10 times with each parameter setting for all the problems in Table 1 (i.e., $3^2 \times 10$ times per problem). The function evaluation number (FEN) and the population size have been set as 100,000 and 100, respectively. The performance metrics (HV, SPREAD, EPSILON, and IGD) are considered as the output of MOEFO for the given parameter setting. While the values of 0.01, 0.5, and 0.99 have been used for α, those of 1, 25, and 50 have been used for K in the DoE methodology. As a result, it is found out that MOEFO performs better with respect to all the metrics on overall when α and K are set to 0.5 and 37, respectively. Hence these values have been set for the experiments in this study.

In the experiments, every algorithm has been run 50 times with a population and generation size of 100 and 1,000 (corresponding to 100,000 FENs), respectively. All the algorithms have been run in parallel using a client machine (Intel Core i7 CPU with 4 cores and 8 threats, 16 GB RAM). The MOEFO implementation in jMetal framework that is ready-to-replicate the same experimentation is shared with the community [7].

5.4.2. Results

This section examines the convergence and diversification performance of every algorithm on the employed benchmark problems by using HV, SPREAD, EPSILON, and IGD metrics. The number of problems on which every algorithm has performed a satisfying performance (with ranking of 1st and 2nd algorithm) has comparatively been shown in Figure 4.

As it is clearly seen in the figure, MOEFO is superior than other algorithms by performing the best/2nd best performance on the problems with respect to HV and EPSILON metrics. The results (the number of problems solved as the best algorithm, the number of problems solved as the second-best algorithm) on HV and EPSILON metrics respectively are (16/15), (11/21). OMOPSO, MOCell, and SPEA2 are the most competitive algorithms against MOEFO with respect to SPREAD and IGD metrics. In addition, it can be seen from the figure that NSGA-II, IBEA, and GDE3 have shown poor performance on overall.

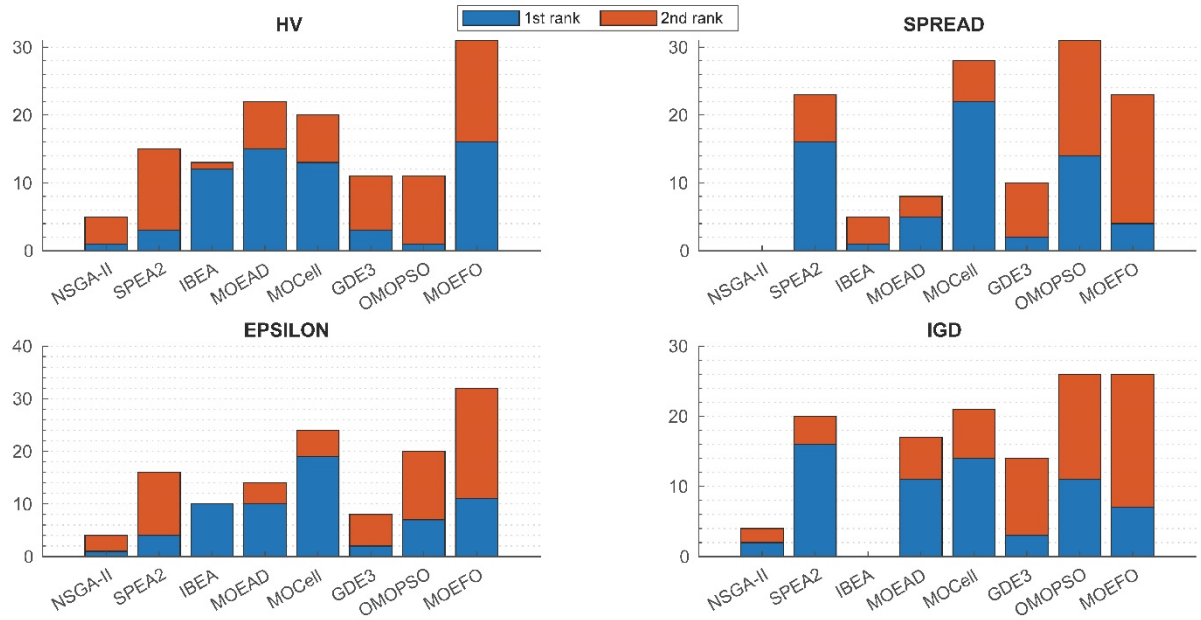

Figure 4. Number of problems every algorithm has shown better performance.

These findings have also been shown separately for every problem family in Figure 5 to reveal if the performances of the algorithms are problem dependent. The figure shows that NSGA-II has shown the poorest performance on all problem families with no exception. While SPEA2 has shown its best performance on DTLZ and MMF problem families, IBEA has shown its best performance on only MMF problem family. However, they perform poorly on the remaining problem families. MOEAD is another algorithm that has shown its better performance on the LZ09 and CEC09 problem families. As for MOCell and GDE3, they become insufficient to solve CEC09, LZ09 problem families. Among them, GDE3 has also shown the poorest performance on MMF problem family. OMOPSO has shown a satisfying performance on overall; but it slightly deteriorates on ZDT family. MOEFO has also shown a better or at least competitive optimization performance on every problem family on overall. However, as compared to MOCell and GDE3 algorithms, MOEFO shows a slight degradation on ZDT, which has only five functions.

To further examine the overall performance of every algorithm, Friedman's ranking test is applied and its results are provided in Table 4. It is worth stressing here that the algorithm with higher mean HV values performs better than others. However, for other metrics, the performance of the algorithm increases as the metric value becomes smaller. The best and 2nd best performing algorithms have been implied with dark and light gray cell colors, respectively. It can be concluded from the table that MOEFO has outperformed its competitors on HV, EPSILON, and IGD metrics. Only on evaluations by using the SPREAD metric, it becomes the third best algorithm and comes after OMOPSO and MOCell. According to these results, the following conclusions can be drawn:

- HV: A much wider search space is dominated by MOEFO algorithm than the competitor algorithms. Moreover, non-dominated solutions found by MOEFO is much closer to the $\mathcal{TPF}^*$ of the problem.
- SPREAD: The distance from the extreme points of non-dominated solutions found by MOEFO to those of $\mathcal{TPF}^*$ is not as close as that yield by OMOPSO and MOCell algorithms.

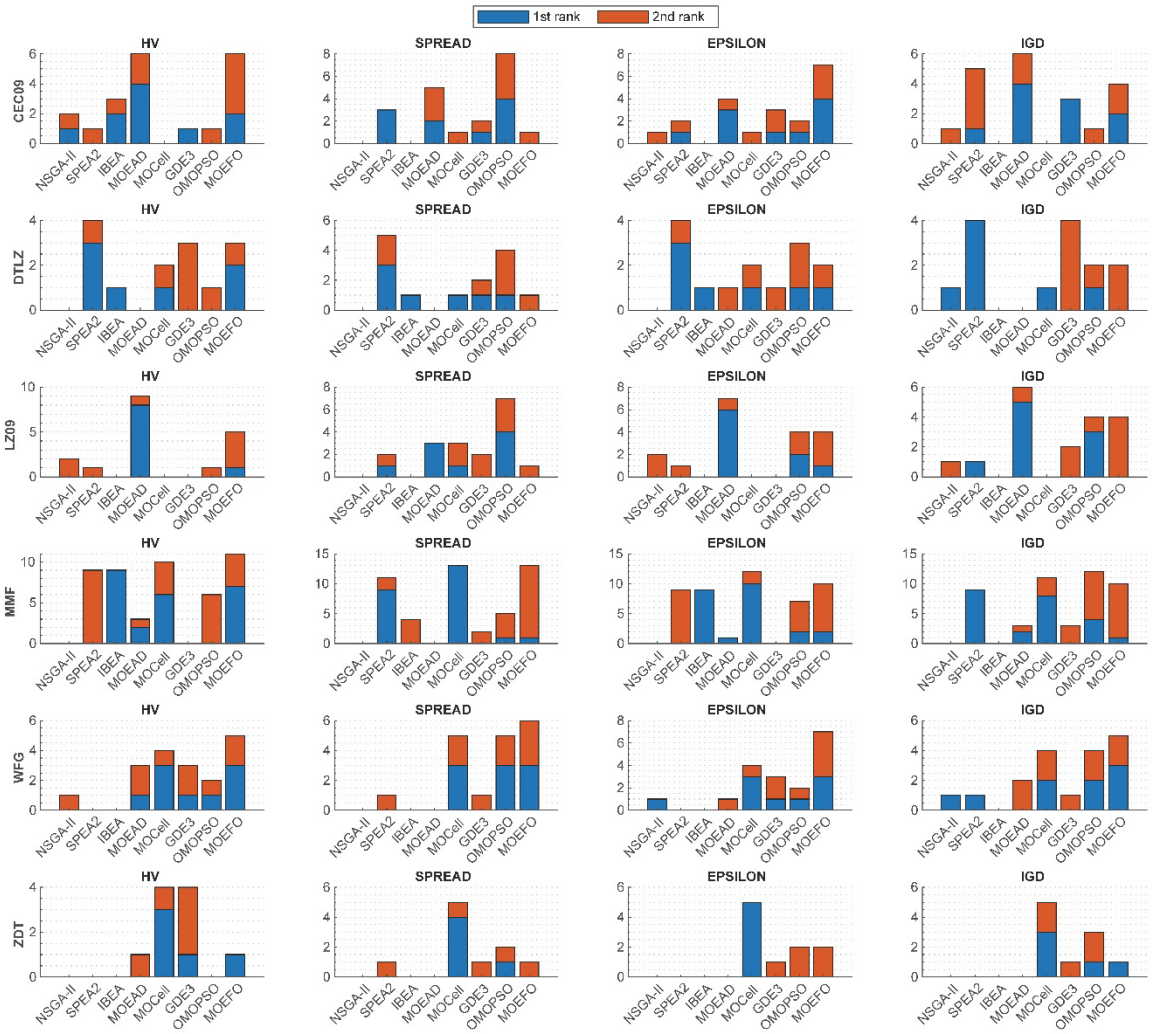

Figure 5. Number of problems every algorithm has shown better performance for every problem family.

- EPSILON: In comparison to other algorithms, MOEFO is able to dominate the solutions in $\mathcal{TPF}^*$ with a less tolerance value on overall. This also supports the finding by HV stating better convergence to $\mathcal{TPF}^*$.
- IGD: The non-dominated solutions found by MOEFO are much diverse and much closer to $\mathcal{TPF}^*$ than all the competitor algorithms.

Wilcoxon signed-rank test at a statistical significance level of 95% has been used for all the performance metrics separately in the experiment in order to find out whether the difference between MOEFO and any of the competitor algorithms is significant or not. In this study, the null hypothesis is that there is no statistically significant difference between the median of the results produced by MOEFO and its competitor algorithm. To speculate that EFO performs much better on a given problem, the null hypothesis should be rejected or an alternative hypothesis should be accepted. The p-values in Wilcoxon's sign test is considered for acceptance of the null hypothesis. So, the null hypothesis can only be rejected as long as the p-value is less than 0.05 (due to the adopted significance level, 95%) and vice versa. Based on the significance comparison, the number of cases i) where null hypothesis is rejected and MOEFO exhibited superior (+) or inferior (-) performance, and ii) where null hypothesis is

accepted and the performance between MOEFO and competitor algorithm is very similar (=) is found. These '+/=/-' cases with respect to every performance metric have been given separately for each problem family in Table 5. The last rows of every problem metric (Sum) show the total count of the cases. The gray cell colors in the table indicate that MOEFO has shown better performance than the competitor algorithms with respect to the evaluation metric. For the sake of better readability, the p-values as well as cumulative positive and negative rank values obtained from Wilcoxon's test have been excluded from the table and they can be found in [7].

Table 4. Friedman's ranking values calculated separately on evaluation metrics for MOOPs.

Algorithms	Ranking			
	HV	**SPREAD**	**EPSILON**	**IGD**
NSGA-II	3.531	6.266	5.406	5.234
SPEA2	4.922	3.703	4.469	3.813
IBEA	3.469	6.438	5.969	7.406
MOEAD	4.453	5.844	4.703	4.578
MOCell	4.719	3.172	4.359	4.281
GDE3	4.898	3.984	3.891	3.766
OMOPSO	4.094	3.016	4.047	3.688
MOEFO	5.914	3.578	3.156	3.324

Similar to the Friedman's ranking evaluation, the Wilcoxon's statistical results also affirm better solutions by MOEFO on HV, EPSILON, and IGD metrics while its performance becomes competitive with respect to the SPREAD metric, where OMOPSO and MOCell have shown their superior performances. From the cumulative performance of the Wilcoxon's test, it can be seen that MOEFO has remarkably shown better performance than all competitor algorithms with respect to the HV, EPSILON, and IGD metrics. It is better than NSGA-II, SPEA2, IBEA, MOEAD, and GDE3 with respect to the SPREAD metric. Only OMOPSO and MOCell algorithms have shown better performance than the proposed MOEFO on only the SPREAD metric according to the Wilcoxon's sign test. To conclude, both Friedman's ranking evaluation and Wilcoxon's sign test results reveal that NSGA-II, IBEA, and MOEAD are the poorest algorithms with respect to these metrics while MOCell and OMOPSO are the most compelling algorithms to MOEFO on overall.

The true Pareto front of problems and the Pareto front of algorithms have been given in Figures 6 and 7 in order to reveal the convergence performance of every algorithm on the objective space. The corresponding box plots that show statistical best, worst, and mean performances have been given in Figure 8. Note that only a subset of problems, in which the proposed MOEFO has become the first algorithm, the second algorithm, and worse than the second algorithm with respect to the HV metric, has been given here. Refer to [7] for all the convergence and box plots that reveal the performance of

Table 5. Comparative number of problems where proposed MOEFO algorithm performs, in order of representation, 'superior(+)/equal(=)/inferior(-)' than the competitors based on the measurement through Wilcoxon signed rank test. (Superior cases are highlighted with gray background color.)

Metric	Problem Family	MOEFO vs.						
		NSGA-II	SPEA2	IBEA	MOEAD	MOCell	GDE3	OMOPSO
HV	CEC09	5/3/2	4/3/3	5/3/2	4/1/5	8/2/0	9/0/1	9/0/1
	DTLZ	4/1/2	2/1/4	5/1/1	5/0/2	2/1/4	3/0/4	7/0/0
	LZ09	6/1/2	6/0/3	7/1/1	0/2/7	8/1/0	8/1/0	8/0/1
	MMF	20/3/1	13/1/10	15/0/9	17/1/6	12/2/10	13/6/5	18/1/5
	WFG	6/2/1	6/2/1	6/3/0	6/0/3	5/0/4	4/1/4	5/1/3
	ZDT	4/0/1	4/0/1	5/0/0	4/0/1	1/0/4	1/0/4	3/0/2
	Sum	45/10/9	35/7/22	43/8/13	36/4/24	36/6/22	38/8/18	50/2/12
SPREAD	CEC09	4/4/2	4/1/5	6/3/1	2/1/7	2/2/6	3/1/6	1/1/8
	DTLZ	7/0/0	2/0/5	5/1/1	7/0/0	3/3/1	2/3/2	1/3/3
	LZ09	4/2/3	3/3/3	5/2/2	3/0/6	2/2/5	1/2/6	1/0/8
	MMF	22/1/1	13/0/11	17/5/2	23/0/1	9/2/13	15/5/4	11/9/4
	WFG	9/0/0	7/0/2	9/0/0	8/1/0	3/2/4	6/1/2	4/3/2
	ZDT	5/0/0	3/1/1	5/0/0	4/1/0	0/0/5	3/0/2	2/0/3
	Sum	51/7/6	32/5/27	47/11/6	47/3/14	19/11/34	30/12/22	20/16/28
EPSILON	CEC09	6/1/3	6/3/1	8/1/1	5/1/4	9/0/1	6/2/2	7/1/2
	DTLZ	2/3/2	2/1/4	5/0/2	5/0/2	2/2/3	2/2/3	4/2/1
	LZ09	5/1/3	5/1/3	7/2/0	1/1/7	7/2/0	6/1/2	4/2/3
	MMF	20/2/2	14/0/10	15/0/9	19/2/3	3/14/7	14/6/4	9/8/7
	WFG	7/2/0	8/1/0	8/1/0	7/1/1	4/3/2	7/0/2	7/1/1
	ZDT	4/0/1	4/0/1	5/0/0	4/0/1	0/0/5	4/0/1	2/1/2
	Sum	44/9/11	39/6/19	48/4/12	41/5/18	25/21/18	39/11/14	33/15/16
IGD	CEC09	5/2/3	4/3/3	10/0/0	4/1/5	9/1/0	7/2/1	9/0/1
	DTLZ	3/2/2	2/1/4	5/1/1	4/1/2	2/2/3	2/2/3	5/2/0
	LZ09	5/1/3	5/0/4	7/2/0	0/2/7	7/1/1	5/1/3	3/2/4
	MMF	20/2/2	13/1/10	24/0/0	19/1/4	8/6/10	15/4/5	5/9/10
	WFG	6/2/1	6/1/2	7/1/1	7/0/2	4/3/2	6/1/2	3/4/2
	ZDT	4/0/1	4/0/1	5/0/0	3/1/1	1/1/3	4/0/1	2/1/2
	Sum	43/9/12	34/6/24	58/4/2	37/6/21	31/14/19	39/10/15	27/18/19

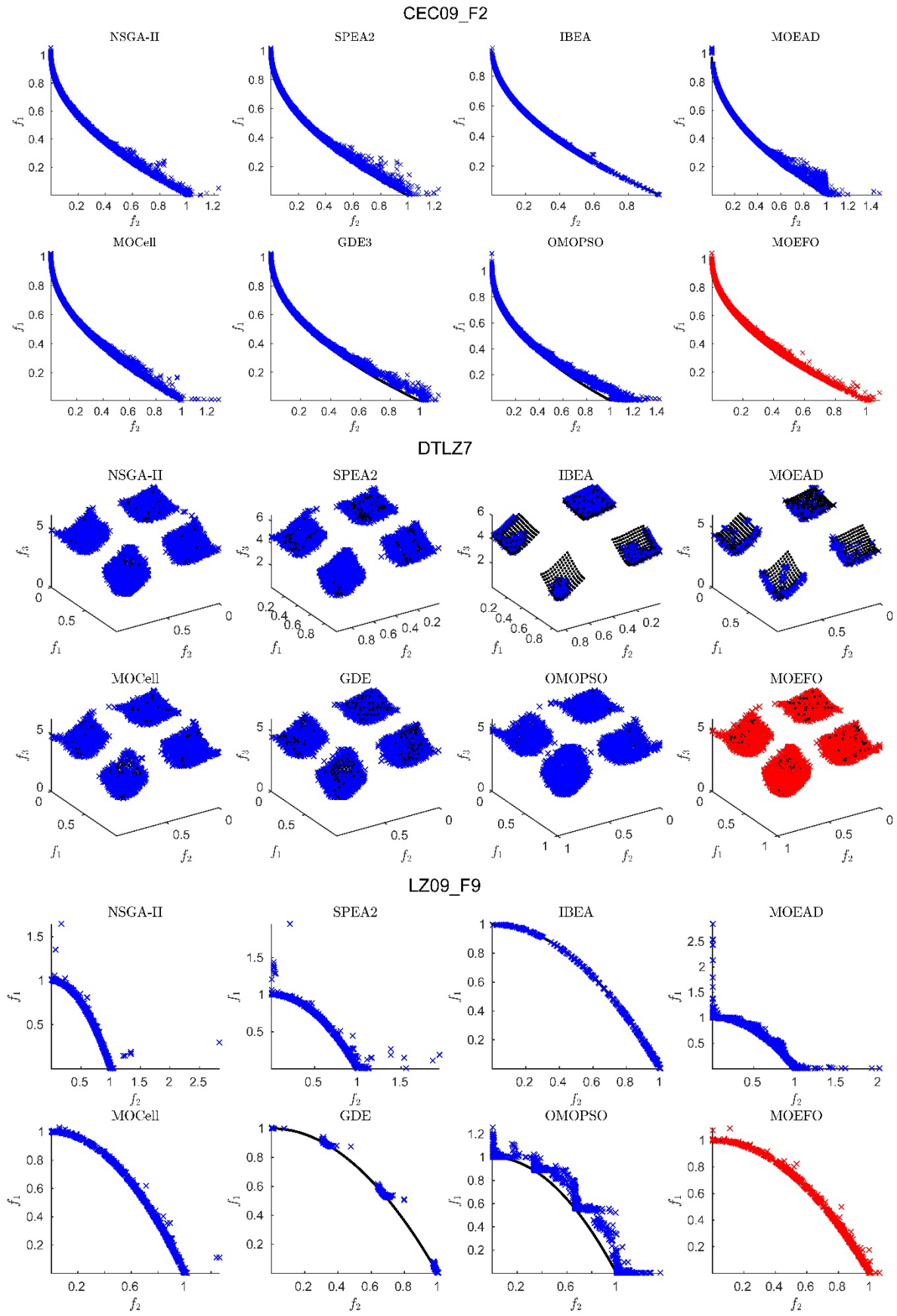

Figure 6. The comparative convergence performance of the algorithms on CEC09_F2, DTLZ7, and LZ09_F9 problems.

every algorithm on each problem by using each metric separately. The approximations to the true Pareto fronts show that MOEFO could competitively represent the true Pareto front of the problems.

Finally, the problem-based and the average run times of the algorithms have been given in Table 6. The results show that CEC09 problem family requires the longest running time due to its higher dimensions (i.e., 30) to be optimized. In addition, it can be seen that OMOPSO requires the least running time while IBEA, MOEAD, and MOCell require the longest running time on overall. The proposed MOEFO algorithm, however, requires an admissible running time for these problem sets on overall. It has shown better performance than the competitors, except for OMOPSO, on CEC09 problem family having higher dimensions, which shows MOEFO can handle high dimensional problems efficiently in terms of run time without sacrificing its optimization performance.

Table 6. Comparative average running times of algorithms for each problem family (in s).

Algorithm	Problem Family					
	CEC09	**DTLZ**	**LZ09**	**MMF**	**WFG**	**ZDT**
NSGA-II	167.41	2.28	3.72	25.33	1.64	2.36
SPEA2	274.77	9.57	6.24	48.20	8.17	8.03
IBEA	278.03	16.43	16.36	56.37	14.91	15.02
MOEAD	258.31	2.37	6.94	112.21	1.12	1.29
MOCell	309.64	5.39	6.53	88.13	4.00	7.20
GDE3	118.14	4.24	1.45	25.14	2.33	2.63
OMOPSO	55.33	2.49	0.89	21.43	0.97	1.18
MOEFO	85.20	4.16	3.76	69.45	3.19	3.62

5.5. Performance Evaluation on MaOP Benchmark Set

5.5.1. Parameter tuning

MOEFO is also explored on MaOPs where their characteristics are summarized in Table 2. To do that, the experimental findings obtained in [46] are taken for comparison in this study. So, the parameter settings used in [46] are adopted for this experimental task. Therefore, MOEFO is run 20 times with 500,000 FENs. While the number of objectives is set to 5, the number of variables to be optimized is set to 10. In [46], NSGA-III and MOEAD algorithms are included for comparison. Six settings where the population size differs from each other are employed. The rationale behind is that the population size in NSGA-III depends on the number of the reference points. Speaking concretely, the population size in this algorithm is equal to $\binom{O+H-1}{O-1}$ where O denotes the number of objectives and H denotes the spaces between reference points in the space. While O is set to 5, the H value ranges from 5 to 10 in [46]. For example, the population size becomes $\frac{(5+10-1)!}{(10)!(5-1)!} = 1001$, when $H = 10$. Here, only the IGD metric is considered for performance assessment as done in [46].

5.5.2. Results

The comparative results on MaOPs are given in Table 7. As stated earlier, proposed MOEFO is run separately for every population setting (the settings are labeled to be a through f in the table). The detailed results are also included in [7]. Every row of the columns a through f in Table 7 indicates the average IGD metric values obtained from 20 runs. The end column, however, gives the average IGD metric values calculated from the 6 populations settings conducted. From these IGD metric results, it

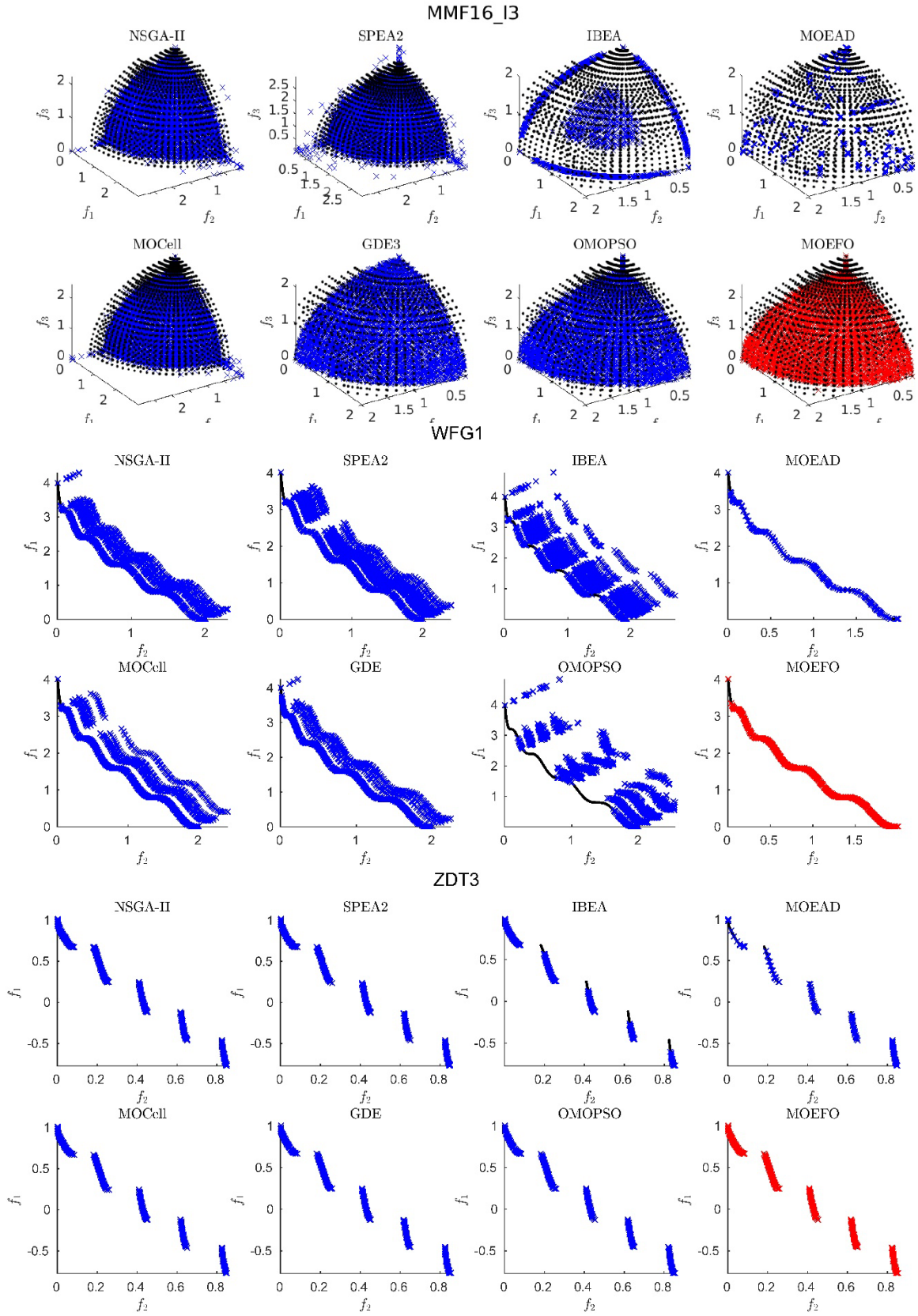

Figure 7. The comparative convergence performance of algorithms on MMF16_l3, WFG1, and ZDT3 problems.

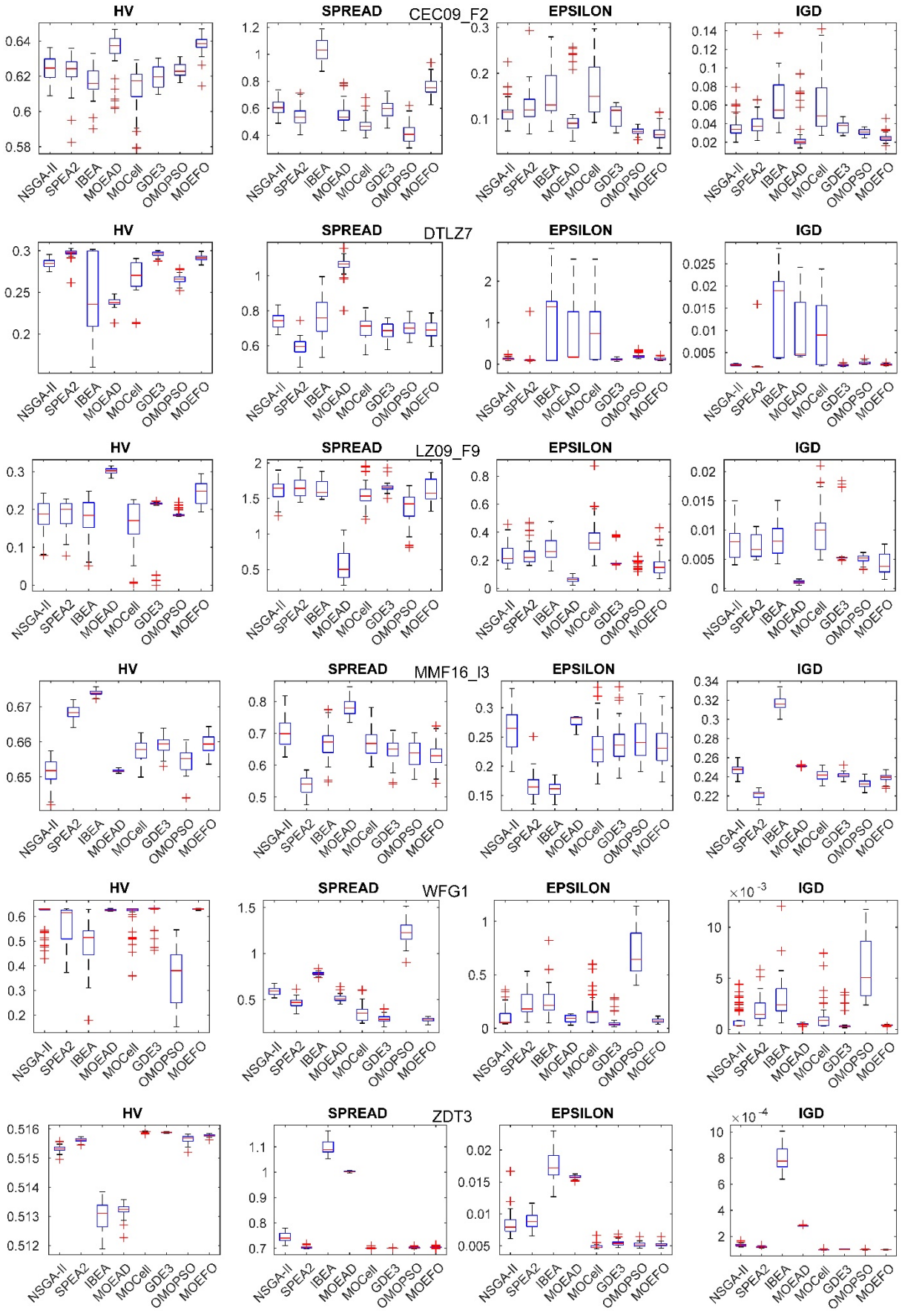

Figure 8. The comparative box plots on CEC09_F2, DTLZ7, LZ09_F9, MMF16_L3, WFG1, and ZDT3.

can be seen that the proposed MOEFO is ranked as the first algorithm on MaOP1, MaOP3, MaOP4, and MaOP5 on overall, while it is ranked as the second algorithm on MaOP2 and MaOP6 problems where NSGA-III and MOEAD, respectively, are ranked as the first algorithm. As for the following problems (i.e., MaOP7 through MaOP10), MOEAD is ranked the first; whereas NSGA-III and MOEFO become very competitive to each other; but NSGA-III performs slightly better and thus is ranked as the second algorithm on overall. When considering the challenges of these problems, it can be concluded that MOEFO is more eligible to solve problems having multimodal, complicated PS, bias, or degeneracy characteristics; but it needs an additional effort to better cope with the problems with local degeneracy characteristic.

Table 7. Comparative average IGD values of MOEFO, MOEAD, and NSGA-III algorithms under six different population size settings (a = 126, b= 220, c = 330, d = 495, e = 715, and f = 1001) for MaOP problems.

Problems	Algorithms	Population Settings						Average
		a	b	c	d	e	f	
MaOP1	MOEFO	17.98	17.33	17.01	16.82	16.72	16.70	17.09
	MOEAD	18.46	18.56	18.25	18.34	18.34	18.41	18.39
	NSGA-III	19.72	17.82	17.46	17.84	17.84	17.40	17.93
MaOP2	MOEFO	0.18	0.14	0.12	0.11	0.09	0.08	0.12
	MOEAD	0.17	0.16	0.16	0.15	0.14	0.14	0.15
	NSGA-III	0.11	0.09	0.09	0.08	0.08	0.08	0.09
MaOP3	MOEFO	0.29	0.25	0.23	0.21	1.12	2.21	0.72
	MOEAD	6.22	7.39	11.65	13.94	16.10	17.93	12.21
	NSGA-III	10.82	13.62	16.27	17.94	19.06	20.65	16.39
MaOP4	MOEFO	0.39	0.38	0.38	0.37	0.37	0.36	0.38
	MOEAD	0.47	0.46	0.45	0.43	0.41	0.40	0.44
	NSGA-III	0.39	0.38	0.38	0.37	0.37	0.36	0.38
MaOP5	MOEFO	0.26	0.25	0.25	0.25	0.25	0.25	0.26
	MOEAD	0.41	0.35	0.31	0.28	0.26	0.23	0.31
	NSGA-III	0.47	0.46	0.46	0.45	0.45	0.43	0.45
MaOP6	MOEFO	0.58	0.55	0.55	0.49	0.46	0.43	0.51
	MOEAD	0.31	0.26	0.20	0.17	0.15	0.12	0.20
	NSGA-III	1.19	0.69	0.71	0.74	0.60	0.51	0.74
MaOP7	MOEFO	0.54	0.48	0.44	0.40	0.37	0.32	0.42
	MOEAD	0.27	0.25	0.21	0.19	0.17	0.15	0.21
	NSGA-III	0.50	0.42	0.31	0.28	0.24	0.21	0.33
MaOP8	MOEFO	0.44	0.40	0.34	0.31	0.28	0.26	0.34
	MOEAD	0.27	0.23	0.19	0.17	0.15	0.14	0.19
	NSGA-III	0.46	0.37	0.34	0.27	0.25	0.20	0.32
MaOP9	MOEFO	0.45	0.36	0.31	0.27	0.24	0.22	0.31
	MOEAD	0.31	0.27	0.22	0.21	0.19	0.18	0.23
	NSGA-III	0.43	0.36	0.29	0.25	0.21	0.19	0.29
MaOP10	MOEFO	0.43	0.36	0.32	0.29	0.25	0.23	0.31
	MOEAD	0.28	0.26	0.22	0.20	0.18	0.17	0.22
	NSGA-III	0.38	0.30	0.26	0.23	0.20	0.17	0.26

6. Conclusion

In this study, the recently proposed promising EFO algorithm is modified for solving MOOPs. A Pareto-based approach is adopted, a new algorithm called Multi-Objective EFO (MOEFO) is introduced. Not only is EFO adapted to solve MOOPs and to give a set of non-dominated solutions as its output, but also the complexity of the algorithm is reduced for efficiency. The effectiveness of the proposed algorithm is explored on both multi-objective optimization problems (MOOPs) and many-objective optimization problems (MaOPs). While NSGA-II [8], SPEA2 [9], IBEA [10], MOEAD [11], MO-Cell [12], GDE3 [13], and OMOPSO [14] are used for MOOPs comparison; NSGA-III [18] and MOEAD [11] are used for MaOPs comparison. A well-known unconstrained MOOPs and MaOPs are used for fair comparison. The algorithms are compared by using the following four metrics for performance evaluation on MOOPs: HV, SPREAD, EPSILON, and IGD. Friedman's ranking and Wilcoxon signed rank tests have been applied for every metric in order to statistically show overall and problem-wise performance, respectively. The statistical results show that MOEFO has outperformed its competitors on HV, EPSILON, and IGD metrics. Only in comparisons based on the SPREAD metric, it becomes the third best algorithm after OMOPSO and MOCell. As for performance evaluation on MaOPs, only the IGD metric is taken into consideration. The results show that the proposed MOEFO has an outstanding optimization performance on majority of the problems included. However, it needs an additional effort to be further improved for the problems having local-degeneracy characteristics. So, we invite the community to come up with improvement solutions, which shall be seen as a future direction of this study.

References

1. Margarita Reyes-Sierra, CA Coello Coello, et al. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. International journal of computational intelligence research, 2(3):287–308, 2006.
2. Reza Akbari, Ramin Hedayatzadeh, Koorush Ziarati, and Bahareh Hassanizadeh. A multi-objective artificial bee colony algorithm. Swarm and Evolutionary Computation, 2:39 – 52, 2012. ISSN 2210-6502. doi: https://doi.org/10.1016/j.swevo.2011.08.001.
3. Reza Akbari and Koorush Ziarati. Multi-objective bee swarm optimization. International Journal of Innovative Computing Information and Control, 8(1B):715–726, 2012.
4. Selim Yilmaz and Sevil Sen. Electric fish optimization: a new heuristic algorithm inspired by electrolocation. Neural Computing and Applications, 2019. ISSN 1433-3058. doi: 10.1007/s00521-019-04641-8.
5. H. Mühlenbein, M. Schomisch, and J. Born. The parallel genetic algorithm as function optimizer. Parallel Computing, 17(6):619 – 632, 1991. ISSN 0167-8191.
6. L. Darrell Whitley. Cellular genetic algorithms. In Proceedings of the 5th International Conference on Genetic Algorithms, pages 658–, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. ISBN 1-55860-299-2.
7. WISE-LAB. MOEFO source code and experimental results. https://wise.cs.hacettepe.edu.tr/projects/moefo, 2020. [Online; accessed 29-Dec-2021].
8. Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE transactions on evolutionary computation, 6(2):182–197, 2002.

9. Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. TIK-report, 103, 2001.
10. Eckart Zitzler and Simon Künzli. Indicator-based selection in multi-objective search. In Xin Yao, Edmund K. Burke, José A. Lozano, Jim Smith, Juan Julián Merelo-Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiňo, Ata Kabán, and Hans-Paul Schwefel, editors, Parallel Problem Solving from Nature-PPSN VIII, pages 832–842, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-30217-9.
11. H. Li and Q. Zhang. Multi-objective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. IEEE Transactions on Evolutionary Computation, 13(2):284–302, April 2009. doi: 10.1109/TEVC.2008.925798.
12. Antonio J Nebro, Juan J Durillo, Francisco Luna, Bernabé Dorronsoro, and Enrique Alba. Design issues in a multiobjective cellular genetic algorithm. In International Conference on Evolutionary Multi-Criterion Optimization, pages 126–140. Springer, 2007.
13. Saku Kukkonen and Jouni Lampinen. GDE3: The third evolution step of generalized differential evolution. In 2005 IEEE congress on evolutionary computation, volume 1, pages 443–450. IEEE, 2005.
14. Margarita Reyes Sierra and Carlos A. Coello Coello. Improving pso-based multi-objective optimization using crowding, mutation and e-dominance. In Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, Evolutionary Multi-Criterion Optimization, pages 505–519, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
15. James David Schaffer. Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms (Artificial Intelligence, Optimization, Adaptation, Pattern Recognition). PhD thesis, Nashville, TN, USA, 1984. AAI8522492.
16. David E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989. ISBN 0201157675.
17. Eckart Zitzler. Evolutionary algorithms for multi-objective optimization: Methods and applications, 1999.
18. 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 Transactions on Evolutionary Computation, 18(4):577–601, 2014.
19. Jouni Lampinen et al. DE's selection rule for multi-objective optimization. Lappeenranta University of Technology, Department of Information Technology, Tech. Rep, pages 03–04, 2001.
20. J. Kennedy and R. Eberhart. Particle swarm optimization. In Neural Networks, 1995. Proceedings., IEEE International Conference on, volume 4, pages 1942–1948 vol.4, Nov 1995.
21. Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization, 39(3):459–471, 2007. ISSN 1573-2916. doi: 10.1007/s10898-007-9149-x.
22. C. A. Coello Coello and M. S. Lechuga. Mopso: a proposal for multiple objective particle swarm optimization. In Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600), volume 2, pages 1051–1056 vol.2, May 2002. doi: 10.1109/CEC.2002.1004388.
23. Cai Dai, Yuping Wang, and Miao Ye. A new multi-objective particle swarm optimization algorithm based on decomposition. Information Sciences, 325:541 – 557, 2015. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2015.07.018.
24. Antonio J Nebro, Juan José Durillo, Jose Garcia-Nieto, CA Coello Coello, Francisco Luna, and Enrique Alba. SMPSO: A new PSO-based metaheuristic for multi-objective optimization. In 2009

IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM), pages 66–73. IEEE, 2009.

25. K. E. Parsopoulos, D. K. Tasoulis, M. N. Vrahatis, and Key Words. Multiobjective optimization using parallel vector evaluated particle swarm optimization. In Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2004, pages 823–828. ACTA Press, 2004.
26. Xiaohui Hu and R. Eberhart. Multiobjective optimization using dynamic neighborhood particle swarm optimization. In Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600), volume 2, pages 1677–1681, May 2002. doi: 10.1109/CEC.2002.1004494.
27. S.N. Omkar, J. Senthilnath, Rahul Khandelwal, G. Narayana Naik, and S. Gopalakrishnan. Artificial bee colony (abc) for multi-objective design optimization of composite structures. Applied Soft Computing, 11(1):489 – 499, 2011. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2009.12.008.
28. Bahriye Akay. Synchronous and asynchronous pareto-based multi-objective artificial bee colony algorithms. Journal of Global Optimization, 57(2):415–445, Oct 2013. ISSN 1573-2916. doi: 10.1007/s10898-012-9993-1.
29. Avadh Kishor, Pramod Kumar Singh, and Jay Prakash. Nsabc: Non-dominated sorting based multi-objective artificial bee colony algorithm and its application in data clustering. Neurocomputing, 216: 514 – 533, 2016. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2016.08.003.
30. Yi Xiang and Yuren Zhou. A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization. Applied Soft Computing, 35:766 – 785, 2015. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2015.06.033.
31. Yi Xiang, Yuren Zhou, and Hailin Liu. An elitism based multi-objective artificial bee colony algorithm. European Journal of Operational Research, 245(1):168 – 193, 2015. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2015.03.005.
32. Y. Liu, L. Ma, and G. Yang. A survey of artificial bee colony algorithm. In 2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), pages 1510–1515, July 2017. doi: 10.1109/CYBER.2017.8446301.
33. H. R. Hassanzadeh and M. Rouhani. A multi-objective gravitational search algorithm. In 2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks, pages 7–12, July 2010. doi: 10.1109/CICSyN.2010.32.
34. Seyedali Mirjalili, Shahrzad Saremi, Seyed Mohammad Mirjalili, and Leandro dos S. Coelho. Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization. Expert Systems with Applications, 47:106 – 119, 2016. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2015.10.039.
35. Xin-She Yang. Multi-objective firefly algorithm for continuous optimization. Engineering with Computers, 29(2):175–184, 2013.
36. Feng Zou, Lei Wang, Xinhong Hei, Debao Chen, and Bin Wang. Multi-objective optimization using teaching-learning-based optimization algorithm. Engineering Applications of Artificial Intelligence, 26(4):1291 – 1300, 2013. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2012.11.006.
37. Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable test problems for evolutionary multiobjective optimization. In Evolutionary multiobjective optimization, pages 105-145. Springer, 2005.

38. H. Li and Q. Zhang. Multi-objective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. IEEE Transactions on Evolutionary Computation, 13(2):284–302, April 2009.
39. Simon Huband, Philip Hingston, Luigi Barone, and Lyndon While. A review of multiobjective test problems and a scalable test problem toolkit. IEEE Transactions on Evolutionary Computation, 10 (5):477–506, 2006.
40. Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary computation, 8(2):173–195, 2000.
41. Qingfu Zhang, Aimin Zhou, Shizheng Zhao, Ponnuthurai Nagaratnam Suganthan, Wudong Liu, and Santosh Tiwari. Multiobjective optimization test instances for the cec 2009 special session and competition. Technical report, 2008.
42. Jing Liang, Ponnuthurai Suganthan, B Qu, D Gong, and Cai Yue. Problem definitions and evaluation criteria for the cec 2020 special session on multimodal multi-objective optimization. Technical report, 2019.
43. Hui Li, Kalyanmoy Deb, Qingfu Zhang, and P.N. Suganthan. Challenging novel many and multi-objective bound constrained benchmark problems. Technical report, 2017.
44. J. J. Durillo and A. J. Nebro. jMetal: A java framework for multi-objective optimization. Advances in Engineering Software, 42(10):760–771, 2011. ISSN 0965-9978. doi: DOI: 10.1016/j.advengsoft.2011.05.014.
45. J.J. Durillo, A.J. Nebro, and E. Alba. The jMetal framework for multi-objective optimization: Design and architecture. In CEC 2010, pages 4138–4325, Barcelona, Spain, July 2010.
46. Hui Li, Kalyanmoy Deb, Qingfu Zhang, P.N. Suganthan, and Lei Chen. Comparison between moea/d and nsga-iii on a set of novel many and multi-objective benchmark problems with challenging difficulties. Swarm and Evolutionary Computation, 46:104-117, 2019. ISSN 2210-6502. doi: 10.1016/j.swevo.2019.02.0