Appl Intell
DOI 10.1007/s10489-017-1019-8

Grasshopper optimization algorithm for multi-objective optimization problems

Seyedeh Zahra Mirjalili[1] · Seyedali Mirjalili[2] · Shahrzad Saremi[2] · Hossam Faris[3] · Ibrahim Aljarah[3]

Abstract This work proposes a new multi-objective algorithm inspired from the navigation of grass hopper swarms in nature. A mathematical model is first employed to model the interaction of individuals in the swam including attraction force, repulsion force, and comfort zone. A mechanism is then proposed to use the model in approximating the global optimum in a single-objective search space. Afterwards, an archive and target selection technique are integrated to the algorithm to estimate the Pareto optimal front for multi-objective problems. To benchmark the performance of the algorithm proposed, a set of diverse standard multi-objective test problems is utilized. The results are compared with the most well-regarded and recent algorithms in the literature of evolutionary multi-objective optimization using three performance indicators quantitatively and graphs qualitatively. The results show that the proposed algorithm is able to provide very competitive results in terms of accuracy of obtained Pareto optimal solutions and their distribution.

Keywords Optimization

✉ Seyedali Mirjalili
ali.mirjalili@gmail.com

[1] School of Electrical Engineering and Computing, University of Newcastle, Callaghan, NSW 2308, Australia

[2] Institute for Integrated and Intelligent Systems, Griffith University, Nathan, QLD 4111, Australia

[3] Business Information Technology Department, King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan

1 Introduction

Before the existence of humans on this planet, nature has been continuously solving challenging problems using evolution. Therefore, it is reasonable to be inspired from nature for solving different challenging problems. In the field of optimization, in 1977, a revolutionary idea was proposed by Holland when evolutionary concepts in nature were simulated in computers for solving optimization problems [1]. It was that moment when the most well-regarded heuristic algorithm called Genetic Algorithms (GA) [2] came to existence and opened a new way of tackling challenging and complex problems in different fields of study.

The general idea of the GA algorithm is very simple. It mimics selection, re-combination, and mutation of genes in nature. In fact, Darwin's theory of evolution was the main inspiration of this algorithm. In GA, the optimization process starts by creating a set of random solutions as candidate solutions (individuals) for a given optimization problem. Each variable of the problem is considered as a gene and the set of variables is analogous to chromosomes. Similarly to nature, a cost function defines the fitness of each chromosome. The whole set of solutions is considered as a population. When the fitness of chromosomes is calculated, the best chromosomes are randomly selected for creating the next population. In GA, the fittest individuals that have higher probability are selected and participate in creating the next population in a similar way to what happens in nature. The next step is the combination of the selected individuals. In this step, the genes of pairs of individuals are randomly merged to produce new individuals. Eventually, some of the individuals' genes in the population are changed randomly to mimic mutation.

The GA became gradually a dominant optimization technique compared to exact (deterministic) approaches [3]

Published online: 04 August 2017

mainly due to the higher probability of local solutions avoidance [4]. However, the main drawback of GA was the stochastic nature of this algorithm which resulted in finding different solutions in every run. This problem seemed easy to handle with enough number of runs yet a large number of function evaluations for every run was still an issue. Advances in computer hardware, nowadays, are reducing the computational cost of GA significantly. Therefore, GA can be considered as a reliable problem solving technique compared to exact methods.

The success of GA in solving a wide range of problems in science and industry paved the way of proposing new heuristic algorithms. The proposal of well-regarded algorithms such as Ant Colony Optimization (ACO) [5], Particle Swarm Optimization (PSO) [6], Evolution Strategy (ES) [7], and Differential Evolution (DE) [8–10] were the results of the success of GA. This field of study is still one of the most popular in computational intelligence field. In addition to the aforementioned advantage of heuristics, local optima avoidance, there are several other advantages that have contributed in their popularity.

Heuristic algorithms do not need to calculate derivation of a problem to be able to solve it. This is because they are not designed based on gradient descent to find the global optimum. They consider a problem as a black box with a set of inputs and outputs. Their inputs are the variables of the problem and outputs are the objectives. A heuristic search starts with creating a set of random inputs as the candidate solutions for the problem. The search is continued by evaluating each solution, observing the objectives values, and changing/combining/evolving the solutions based on their outputs. These steps are repeated until a termination criterion is met, which can be either a threshold of exceeding maximum number of iterations or maximum number of function evaluations.

Although heuristics are very effective in solving real challenging problems, there are many difficulties when solving optimization problems [11]. In addition, optimization problems are not all similar and have diverse characteristics. Some of these difficulties/features are: dynamicity [12–14], uncertainty [15], constraints, [16], multiple objectives [17], and many objectives [18]. Each of such difficulties has created a subfield in the field of heuristics and attracted many researchers.

In dynamic problems, the position of the global optimum changes over time. Therefore, a heuristic should be equipped with suitable operators to track the changes and to not lose the global optimum [19]. In real problems, there is a variety of uncertainty applied to each component. In order to address this, a heuristic should be able to find robust solutions that are fault tolerant. Constraints are another difficulty of a real problem, which restricts the search space. They divide solutions to feasible and infeasible. This means that heuristics should be equipped with suitable operators to discard infeasible solutions during optimization and eventually find the best feasible solution. There are many outstanding constraint handling techniques in the literature [20, 21]. Since the proposed technique of this work will be applied to only unconstrained problems, we do not review the literature of such techniques further and refer interested readers to [20, 21].

Some problems have computationally expensive objective functions, which make the whole heuristic optimization process very long due to the need to evaluate solutions iteratively. In order to solve such a problem, researchers try to reduce the number of function evaluations or utilize surrogate models, which are computationally much cheaper.

One of the main difficulties mentioned above is the existence of multiple objectives. A heuristic algorithm cannot compare solutions when there is more than one objective to be considered. In order to solve such problems, researchers compared solutions using Pareto dominance operator [22]. Due to the nature of such problems, in addition, there is more than one best (non-dominated) solution for a multi-objective problem. A heuristic should be able to find all the best solutions. Multi-objective optimization using heuristics has recently attracted much attention and is the main focus of this work [23].

The majority of the heuristics have been equipped with proper operators to solve multi-objective problems in the literature. The mechanism of most of multi-objective heuristics is almost identical. The essential component is an archive or repository to store the non-dominated solutions during the optimization process. Multi-objective heuristics iteratively update this archive to improve the quality and quantity of non-dominated solutions in the archive. Another duty of a multi-objective heuristic is to find "different" non-dominated solutions. This means that the non-dominated solutions should be spread uniformly across all objectives to show all the best trade-offs between the multiple objectives. This is a key feature in *a posteriori* multi-objective algorithms [24] where decision making [25] occurs after the optimization.

There are many algorithms in the literature for solving multi-objective algorithms. For the GA, the most well-regarded multi-objective counterpart is Non-dominated Sorting Genetic Algorithm (NSGA) [26]. Other popular algorithms are: Multi-Objective Particle Swarm Optimization (MOPSO) [27–29] Multi-objective Ant Colony Optimization [30], Multi-Objective Differential Evolution [31], Multi-objective Evolution Strategy [32]. All these algorithms are proved to be effective in finding non-dominated solutions for multi-objective problems. However, there question is if we still need more algorithms. According the No Free Lunch theorem for optimization [33], there is no algorithm capable of solving optimization algorithms of

all kinds. This theorem logically proved this and allows the proposal of new algorithms or improvement of current ones. Therefore, there is still room for new or improved algorithms to better solve the current problems and solve problems that are difficult to solve with the current techniques.

Grasshopper Optimization Algorithm (GOA) [40] has been proven to benefit from high exploration while showing very fast convergence speed. The special adaptive mechanism in this algorithm smoothly balances exploration and exploitation. These characteristics make the GOA algorithm potentially able to cope with the difficulties of a multi-objective search space and outperform other techniques. In addition, the computational complexity is better than those of many optimization techniques in the literature. These powerful features motivated our attempts to propose a multi-objective optimizer inspired from the social behaviour of grasshoppers in nature. The rest of this paper is organized as follows:

Section 2 provides the preliminaries, essential definition, and literature review of multi-objective optimization using heuristics. The GOA and its proposed multi-objective version are described in detail in Section 3. The results are presented, discussed, and analysed in Section 4. The latter section also includes the experimental set up, test functions, and performance metrics as well. Finally, Section 5 concludes the work and suggests several future research directions.

2 Multi-objective optimization

2.1 Preliminaries and definitions

As its name implies, multi-objective optimization deals with optimizing multiple objectives. In the literature, the term multi-objective refers to problems with up to four objectives. Due to the complexity of problems with more than 4 objectives, there is a specialized field called many-objective optimization to solve problems with many objectives. Since solving such problems is out of the scope of this work, interested readers are referred to a survey in [34].

In order to formulate a multi-objective problem, we can use a problem definition. Without the loss of generality, the following equations show multi-objective optimization as a minimization optimization problem [35]:

$$\begin{aligned} &Minimize: && F(\vec{x}) = \{f_1(\vec{x}), f_2(\vec{x}), \ldots, f_0(\vec{x})\} && (2.1)\\ &Subject\ to: && g_i(\vec{x}) \geq 0, \quad i = 1, 2, \ldots, m && (2.2)\\ &&& h_i(\vec{x}) = 0, \quad i = 1, 2, \ldots, p && (2.3)\\ &&& L_i \leq U_i \qquad , i = 1, 2, \ldots, n && (2.4) \end{aligned}$$

where n is the number of variables, o is the number of objective functions, m is the number of inequality constraints, p is the number of equality constraints, g_i is the *i-th* inequality constraints, h_i indicates the *i-th* equality constraints, and $[Li, Ui]$ are the boundaries of *i-th* variable.

One of the main difficulties in a multi-objective search space is that objectives can be in conflict and require special considerations. In a multi-objective search space, a solution cannot be compared with another with relational operators. This is due to the existence of more than one criterion for comparison [36]. Therefore, we need other operators to measure and find out how much a solution is better than another. The most widely used operator is called Pareto dominance and defined as follows:

$$\forall i \in \{1, 2, \ldots, 0\} : f_i(\vec{x}) \leq f_i(\vec{y}) \wedge \exists i \in \{1, 2, \ldots, k\} : f_i(\vec{x}) \leq f_i(\vec{y}) \quad (2.5)$$

where $\vec{x} = (x_1 x_2, \ldots, x_k)$ and $\vec{y} = (y_1 y_2, \ldots, y_k)$.

This equation shows that a solution (vector x) is better than another (vector y) if it has equal and at least one better value on all objectives. In this case, x is said to dominate y and it can be denoted as: $x \prec y$. An example is presented in Fig. 1. This figure shows that the circles are better than squares because they provide a lower value in both objectives.

Despite the fact that circles dominated the squares in the above figures, they are non-dominated with respect to each other. This means that they are better in one objective and worse in the other. The Pareto optimality can be mathematically defined as follows:

$$\forall i \in \{1, 2, \ldots, 0\} : \{\nexists \vec{y} \in X \,|\, f_i(\vec{y}) \prec f_i(\vec{x})\} \quad (2.6)$$

This solution is referred to as a Pareto optimal solution because it cannot be dominated by the solution x.

For every problem, there is a set of best non-dominated solutions. This set is considered as a solution for multi-objective optimization. Consequently, the projection of Pareto optimal solutions in the objective space are stored in a set called Pareto optimal front.

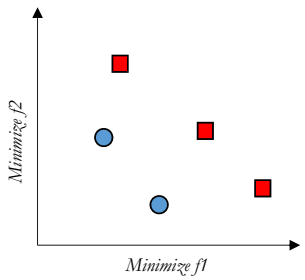

Fig. 1 Pareto dominance

2.2 Multi-objective optimization using metaheuristics

In the literature, there are three main approaches for solving multi-objective problems using metaheuristics: *a priori* [37], *a posteriori* [38], and interactive [39]. In the first approach, multiple objectives are aggregated to one objective. This means that the multi-objective problem is converted to a single-objective as follows:

$$Minimize: \quad F(\vec{x}) = w_1 f_1(\vec{x}) + w_2 f_2(\vec{x}) + \ldots + w_o f_o(\vec{x}) \tag{2.7}$$

$$Subject\ to: \quad g_i(\vec{x}) \geq 0, \qquad i = 1, 2, \ldots, m \tag{2.8}$$

$$h_i(\vec{x}) = 0, \qquad i = 1, 2, \ldots, p \tag{2.9}$$

$$L_i \leq x_i \leq U_i, \quad i = 1, 2, ..., n \tag{2.10}$$

where w_1 , w_2, w_3,, ..., w_o are the weights of objectives, n is the number of variables, o is the number of objective functions, m is the number of inequality constraints, p is the number of equality constraints, g_i is the *i-th* inequality constraints, h_i indicates the *i-th* equality constraints, and *[Li,Ui]* are the boundaries of *i-th* variable.

Aggregation of objectives allows single-objective optimizers to effectively find Pareto optimal solutions. However, the main drawbacks of this approach are the need to run an algorithm multiple times to find multiple Pareto optimal solutions, dealing with all the challenges in every run, lack of information exchange between Pareto optimal solutions during optimization, the need to consult with an expert to find the best weights, and the failure to find concave regions of Pareto optimal front due to the addition of objectives. Such methods are called *a priori* because decision making is done before the optimization within determining the weights. Obviously, the disadvantages of *a priori* approaches outweigh their benefits. The main duty of a designer when using such techniques is to run the algorithm multiple times while changing the weights to find the Pareto optimal front.

The second class of multi-objective metaheuristics are considered as *a posteriori*. As the name implies, decision making is done after the optimization. There is no aggregation anymore and such methods maintain the multi-objective formulation of a multi-objective problem. The main advantages of this class are the ability of finding Pareto optimal solution set in just one run, exchanging information between Pareto optimal solutions during optimization, and the ability to determine the Pareto optimal front of any type. However, *a posteriori* methods require special mechanisms to address multiple and often conflicting objectives. In addition, the computational cost of such methods is normally higher than that of aggregation techniques.

The last method mentioned above is called interactive multi-objective optimization. The name indicates that decision making is done during optimization. Interactive optimization is also called human-in-the-loop optimization, in which an expert preference is continuously fetched and involved during the optimization process to find desired Pareto optimal solutions.

The literature shows that *a posteriori* are the dominant methods of multi-objective optimization. Most of the well-regarded algorithms in the field of single-objective optimization have been modified to perform a posteriori multi-objective optimization. Needless to say, they all compare solutions based on Pareto dominance and employ an archive to store the best Pareto optimal solutions obtained so far. The general framework of all *a posteriori* methods is identical. They initiate the optimization process with a set of random solutions. After finding the Pareto optimal solutions and storing them to an archive, they try to improve the solutions to find better Pareto optimal solutions. The process of improving Pareto optimal solutions is stopped when a condition is met.

The main objective of *a posteriori* multi-objective algorithm is to find a very accurate approximation of the actual (true) Pareto optimal solutions for a given multi-objective problem. Due to the occurrence of decision making after the optimization proves, the solutions should be spread along all objectives as uniform as possible as well. One of the main challenges here is that finding accurate Pareto optimal solutions (convergence) is in conflict with distribution of solutions (coverage). A multi-objective optimization should be able to effectively balance these two factors to solve a multi-objective problem.

In order to improve the coverage, different mechanisms are utilized. In MOPSO, for instance, Pareto optimal solutions in the less populated segments of the archive have a higher probability to be chosen as the leaders for other particles. In NSGA-II, non-dominated sorting ranks Pareto optimal solutions and assigns them a number. This gives better Pareto optimal solutions a higher chance of participation in creating the new generation.

In spite of the recent advances in the field of multi-objective optimization, many researchers try to improve the current techniques or propose new ones to solve the current multi-objective optimization problems better. This motivated our attempts to proposed and investigate the effectiveness of a new algorithm called Grasshopper Optimization Algorithm (GOA) in this field. In the next section, the GOA is introduced first and then the multi-objective version of this algorithm is proposed.

3 Multi-objective grasshopper optimization algorithm (MOGOA)

This section first introduces the GOA algorithm. The multi-objective version of this algorithm is then proposed.

3.1 Grasshopper optimization algorithm

The GOA algorithm simulates the swarming behaviour of grasshoppers in nature. The mathematical equations and formulas proposed for this algorithm are given as follows [40, 41]. In GOA, the position of the grasshoppers in the swarm represents a possible solution of a given optimization problem. The position of the *i-th* grasshopper is denoted as X_i and represented as given in (3.1)

$$X_i = S_i + G_i + A_i \tag{3.1}$$

where S_i is the social interaction, G_i is the gravity force on *i-th* grasshopper, and A_i shows the wind advection.

Equation (3.1) includes three main components to simulate social interaction, impact of gravitational force, and wind advection. These components fully simulate the movement of grasshoppers, yet the main component originated from grasshoppers themselves is the social interaction discussed as follows:

$$S_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} s(d_{ij})\widehat{d_{ij}} \tag{3.2}$$

where d_{ij} is the distance between *i-th* and *j-th* grasshopper and it is calculated as $d_{ij} = |x_j - x_i|$, s is a function to define the strength of social forces as shown in (3.3), and $\widehat{d_{ij}} = \frac{x_j - x_i}{d_{ij}}$ is a unit vector from *i-th* grasshopper to the *j-th* grasshopper.

The s function, which defines the social forces, is calculated as follows:

$$s(r) = fe^{\frac{-r}{l}} - e^{-r} \tag{3.3}$$

where f indicates the intensity of attraction and $l$is the attractive length scale.

The function s is illustrated in the following figure to show how it impacts the social interaction (attraction and repulsion) of grasshoppers.

Inspecting Fig. 2, it may be seen that repulsion forces are encouraged in the interval of [0, 2.079]. If the distance becomes equal to 2.079, there is no attraction and repulsion. This area is called comfort area. The attraction force increases from 2.079 unit of distance to nearly 4 and then it gradually decreases. Changing the parameters l and f in (3.3) results in different social behaviours in artificial grasshoppers as may be seen in Fig. 3.

To show the interaction between grasshoppers with respect to comfort area, Fig. 4 shows a conceptual schematic.

Despite the merits of the function s, it is not able to apply strong forces between grasshoppers with large distances between them. To resolve this issue, the distance between grasshoppers should be mapped or normalized to [1, 4].

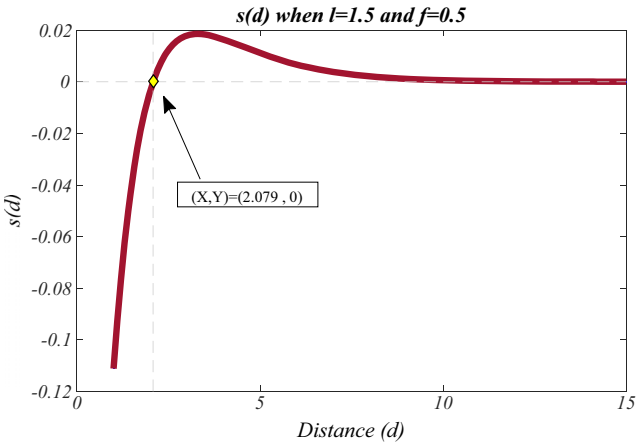

Fig. 2 Function s when l =1.5 and f =0.5

The G component in (3.1) is calculated as follows:

$$G_i = -g\hat{e}_g \tag{3.4}$$

where g is the gravitational constant and $\hat{e}_g$ shows a unity vector towards the centre of earth.

The A component in (3.1) is calculated as follows:

$$A_i = u\hat{e_w} \tag{3.5}$$

where u is a constant drift and $\hat{e_w}$ is a unity vector in the direction of wind.

Equation (3.1) can be written with all components as follows:

$$X_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} s\left(|x_j - x_i|\right)\frac{x_j - x_i}{d_{ij}} - g\hat{e}_g + u\hat{e_w} \tag{3.6}$$

where $s(r) = fe^{\frac{-r}{l}} - e^{-r}$ and N is the number of grasshoppers.

To solve optimization problems, a stochastic algorithm must perform exploration and exploitation effectively to determine an accurate approximation of the global optimum. The mathematical model presented above should be equipped with special parameters to show exploration and exploitation in different stages of optimization. The proposed mathematical model is as follows:

$$X_i^d = c\left(\sum_{\substack{j=1 \\ j \neq i}}^{N} c\frac{ub_d - lb_d}{2} s\left(\left|x_j^d - x_i^d\right|\right)\frac{x_j - x_i}{d_{ij}}\right) + \hat{T}_d \tag{3.7}$$

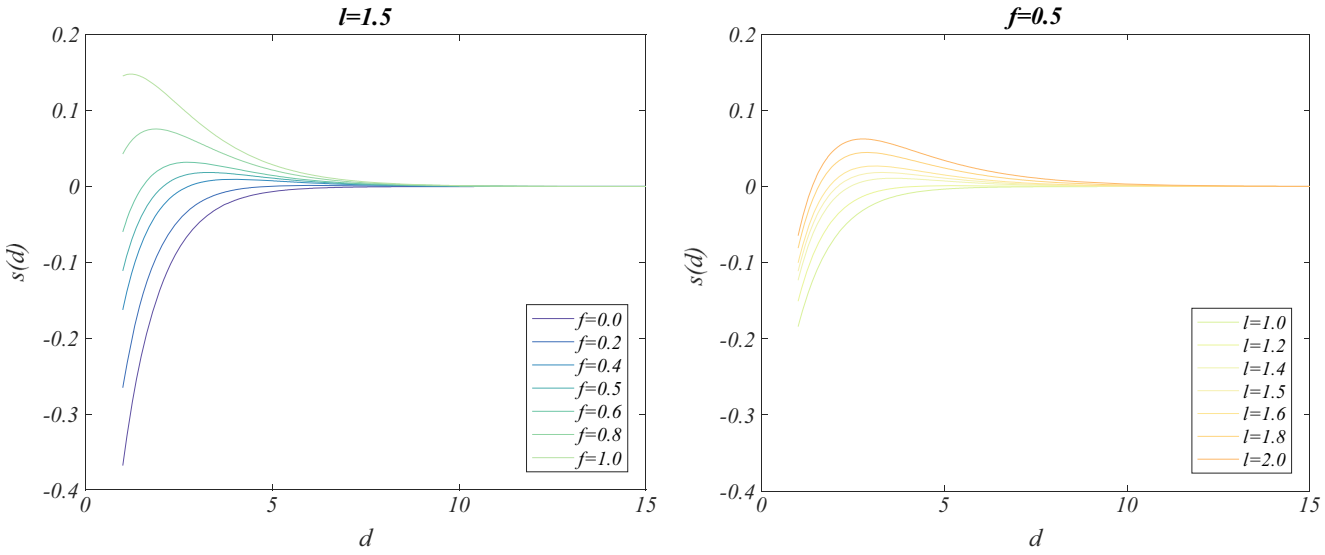

Fig. 3 Behaviour of the function s when varying l or f

where ub_d is the upper bound in the d-th dimension, lb_d is the lower bound in the d-th dimension $s\,(r) = f e^{\frac{-r}{l}} - e^{-r}$, $\hat{T}_d$ is the value of d-th dimension in the target (best solution found so far), and c is a decreasing coefficient to shrink the comfort area, repulsion area, and attraction area. Note that S is almost similar to the S component in (3.1). However, we do not consider the gravity (no G component) and assume that the wind direction (A component) is always towards a target ($\hat{T}_d$).

It should be noted that the inner c contributes to the reduction of repulsion/attraction forces between grasshoppers proportional to the number of iterations, while the outer c

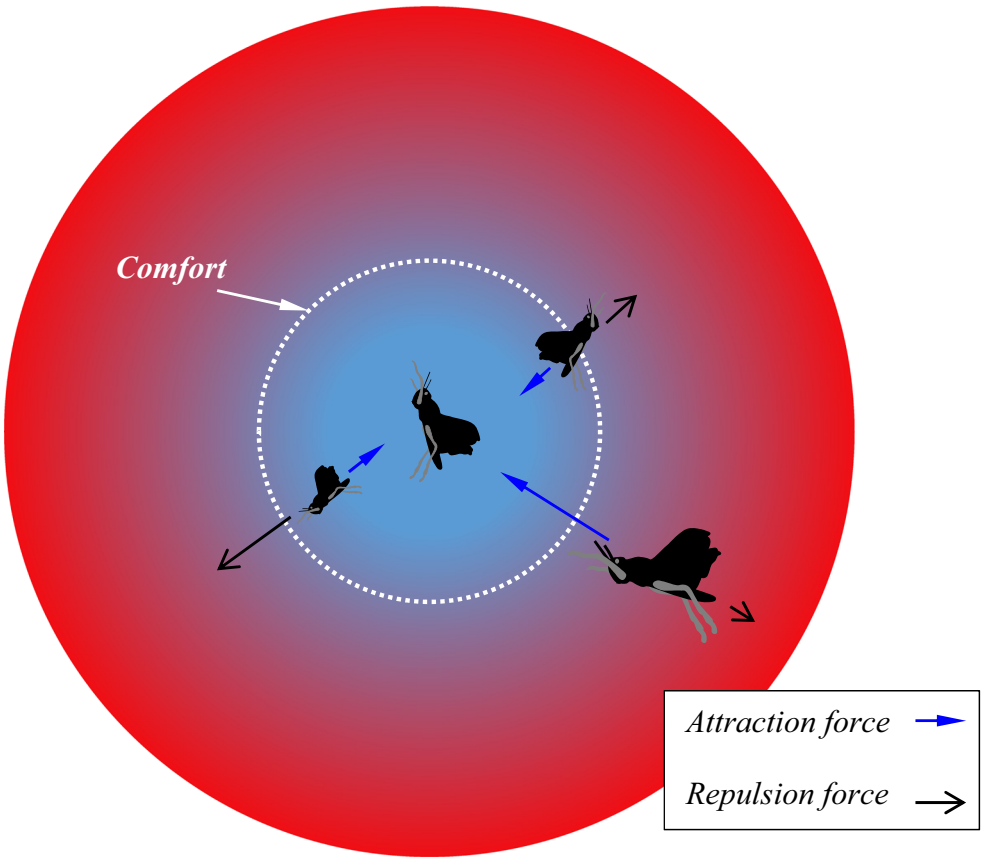

Fig. 4 Primitive corrective patterns between individuals in a swarm of grasshoppers

reduces the search coverage around the target as the iteration counter increases.

The parameter c is updated with the following equation to reduce exploration and increase exploitation proportional to the number of iteration:

$$c = cmax - l\frac{cmax - cmin}{L} \tag{3.8}$$

where *cmax* is the maximum value, *cmin* is the minimum value, l indicates the current iteration, and L is the maximum number of iterations. In this work, we use 1 and 0.00001 values for *cmax* and *cmin,* respectively.

3.2 Multi-objective grasshopper optimization algorithm (MOGOA)

A multi-objective algorithm seeks two goals, when solving multi-objective problems. For one, a very accurate approximations of the true Pareto optimal solutions should be found. For another, the solutions should be well-distributed across all objectives. This is essential in *a posteriori* methods since the decision making is done after the optimization process. In the following paragraphs, the main mechanisms to achieve these two essential goals are discussed.

As discussed in the preceding section, two solutions cannot be compared with the regular relational operators. Also, there is more than one solution for a multi-objective problem. In order to compare the solutions in MOGOA, Pareto optimal dominance is used. The best Pareto optimal solutions are also stored in an archive. The main challenge in designing MOGOA based on GOA is to choose the target. The target is the main component that leads the search agents towards promising regions of the search space. The same equations in the preceding section are used in the MOGOA, and the main difference is the process of updating the target.

The target can be chosen easily in a single-objective search space by choosing the best solution obtained so far. However, the target should be chosen from a set of Pareto optimal solutions in MOGOA. Obviously, the Pareto optimal solutions are added to the archive and the target must be one of them in the archive. The challenge here is to find a target to improve the distribution of solutions in the archive. In order to achieve this, the number of neighbouring solutions in the neighbourhood of every solution in the archive is first calculated considering a fixed distance. This approach is similar to that in MOPSO. Afterwards, the number of neighbouring solutions is counted and assumed as the quantitative metric to measure the crowdedness of regions in the Pareto optimal front. The following equation is employed which defines the probability of choosing the target from the archive:

$$P_i = 1 \Big/ N_i \tag{3.9}$$

where N_i is the number of solutions in the vicinity of the *i-th* solution.

With this probability, a roulette wheel is utilized to select the target from the archive. This allows improving the distribution of less distributed regions of the search space. Another advantage is that in case of premature convergence, it is possible for solutions with crowded neighbourhood to be selected as the target and resolve this issue.

The archive that is employed has a limitation. There should be a limited number of solutions in the archive to be able to decrease the computational cost of MOGOA. This leads to the issue of a full archive. We deliberately remove solutions with crowded neighbourhood to decrease the number of solutions in the crowded regions. This allows accommodation of new solutions in the less populated regions. In order to do so, the inverse of the P_i in (4.1) and a roulette wheel are used.

The archive should also be updated regularly. However, there are different cases when comparing a solution outside the archive and the solutions inside the archive. The MOGOA should be able to handle those cases to improve the archive. The easiest case is where the external solution is dominated by at least one of the archive members. In this case it should be thrown away immediately. Another case is when the solution is non-dominated with respect to all solutions in the archive. Since the archive stores non-dominated solutions obtained so far, a non-dominated solution should be added to the archive. However, if the solution dominates a solution in the archive, it should be replaced with it.

The computational complexity of MOGOA is of $O(MN^2)$ where M is the number of objectives and N is the number of solutions. The complexity is equal to other well-known algorithms in this field: NSGA-II [42], MOPSO, SPEA2 [43], and PAES [7]. The computational complexity is better than other algorithms such as NSGA [44] and SPEA [45], which are of $O(MN^3)$.

Note that the MOGOA algorithm was designed above considering the 'Unified Framework' proposed by Padhye et al. [46, 47] in which the main steps are initialization, selection, generation and replacement. Improving the performance of MOGOA with integrating evolutionary operators (e.g. crossover and mutation) is out of the scope of this work, but it would be a valuable contribution in future.

With the above-mentioned mechanisms and rules, the MOGOA is able to find the Pareto optimal solutions, store them in the archive, and improve their distribution. In the following section, a set of test functions is employed to test the performance of the proposed MOGOA algorithm.[1]

[1] The source codes of MOGOA can be found at http://www.alimirjalili.com/Projects.html.

4 Results on test functions

4.1 Experimental set up

Similarly to benchmarking single-objective optimization algorithms, several test functions with diverse characteristics should be employed. This is because different test functions are able to challenge an algorithm from different perspectives. There are many standard test functions in the literature: ZDT [48], DTLZ [49], and CEC2009 [50]. The details of test functions employed in this work are presented in the Appendix. It can be seen that the test functions have different Pareto optimal front: concave, convex, linear, and separated. The search space of most of them are multimodal, in which several local fronts hinder the solutions to move easily towards the true Pareto optimal front.

In order to verify the results of the proposed MOGOA algorithm, its results are compared with the well-regarded and popular multi-objective algorithms in the literature such as NSGA-II, MOPSO, MODA [51], and MOALO [52]. The results are collected and compared qualitatively and quantitatively. For the qualitative results, the best Pareto optimal fronts over 10 runs are chosen and drawn in the following subsection. Obviously, such figures allow us to see which algorithm performs better. However, qualitative figures cannot show how much better an algorithm is better accurately. Therefore, we have employed a set of performance indicators to quantify convergence and coverage of the algorithms. The first performance metric employed is the inverse generational distance (IGD) and defined as follows:

$$IGD = \frac{\sqrt{\sum_{i=1}^{nt} \left(d_i'\right)^2}}{n} \tag{4.1}$$

where nt is the number of true Pareto optimal solutions and d_i' indicates the Euclidean distance between the *i-th* true Pareto optimal solution and the closest Pareto optimal solution obtained in the reference set.

The IGD metric quantifies the convergence of algorithms, so we can measure how close the obtained Pareto optimal solutions are to the true Pareto optimal solutions. However, the coverage of solutions across all objectives is also important. To measure coverage and quantitatively compare the algorithms, the spacing (SP) and maximum spread (MS) measures are employed. SP and MS are given in (4.2) and (4.3), respectively.

$$\mathrm{SP} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} \left(\bar{d} - d_i\right)^2} \tag{4.2}$$

where $\bar{d}$ is the average of all d_i, n is the number of Pareto optimal solutions obtained, and $d_i = \left(\left|f_1^i(\vec{x}) - f_1^j(\vec{x})\right| + \left|f_2^i(\vec{x}) - f_2^j(\vec{x})\right|\right)$ for all $i, j = 1,2,3,\ldots,n$.

$$\mathrm{MS} = \sqrt{\sum_{i=1}^{o} \max\left(d\left(a_i, b_i\right)\right)} \tag{4.3}$$

where d is a function to calculate the Euclidean distance, a_i is the maximum value in the *i-th* objective, b_i is the minimum in the*i-th* objective, and o is the number of objectives.

Note that for IGD and SP, lower values show better results. By contrast, MS is high for a better algorithm and shows higher coverage. In order to qualitatively compare the results and observe the distribution of solutions, the best Pareto optimal front by each of the algorithms are illustrated in the following subsection as well.

4.2 Results on ZDT test suite

The results of algorithms on the test functions are presented in Table 1 and Fig. 5. Table 1 includes average, standard deviation, median, best, and worst value for IGD obtained by algorithms after 10 independent runs. Figure 5 illustrates the best Pareto optimal front obtained by all algorithms.

Inspecting the results in Table 1, it is evident that the MOGOA algorithm outperforms MOPSO and NSGA-II in three of the test functions: ZDT13, ZDT1 with linear front, and ZDT1 with 3 objectives. In the rest of the test functions, MOGOA shows very competitive results. It is interesting that MOGOA managed to significantly show better results than NSGA-II in all these functions. The IGD performance measure is a good indicator of the accuracy of algorithms in approximating Pareto optimal solutions. Therefore, these results show that MOGOA benefits from good convergence towards the Pareto optimal solutions.

The best Pareto optimal fronts illustrated in Fig. 5 show that the distribution of the Pareto optimal solutions obtained by MOGOA tends to be better than the other two algorithms. It may be seen that MOGOA outperforms NSGA-II in all the functions and is occasionally better than MOPSO. The Pareto optimal fronts for ZDT3 indicate that MOGOA finds four separated regions, while MOPSO finds only three of them. In the most challenging test function, ZDT1 with 3 objectives, it can be observed that the Pareto optimal solutions found by MOGOA are well distributed.

4.3 Results on CEC2009 test suite

The preceding section investigated the performance of the proposed MOGOA algorithm on ZDT test set. Most of

Table 1 Results of the multi-objective algorithms on ZDT1, ZDT2, ZDT3, ZDT1 with linear front, and ZDT1 with 3 objectives

Algorithm	IGD				
	Ave	Std.	Median	Best	Worst
ZDT1					
MOGOA	0.0121	0.0247	0.0046	0.0028	0.0822
MOPSO	0.0042	0.0031	0.0037	0.0015	0.0101
NSGA-II	0.0599	0.0054	0.0574	0.0546	0.0702
MODA	0.0061	0.0028	0.0072	0.0024	0.0096
MOALO	0.0152	0.00502	0.0166	0.0061	0.0209
ZDT2					
MOGOA	0.007	0.0090	0.0049	0.0016	0.0273
MOPSO	0.003	0.0002	0.0017	0.0013	0.0017
NSGA-II	0.140	0.0263	0.1258	0.1148	0.1834
MODA	0.017	0.0109	0.0165	0.0050	0.0377
MOALO	0.017	0.0109	0.0165	0.0050	0.0377
ZDT3					
MOGOA	0.0306	0.0034	0.0313	0.0224	0.0345
MOPSO	0.0378	0.0063	0.0362	0.0308	0.0497
NSGA-II	0.0417	0.0081	0.0403	0.0315	0.0557
MODA	0.0279	0.0040	0.0302	0.02	0.0304
MOALO	0.0303	0.0009	0.0323	0.0303	0.0330
ZDT1 with a linear front					
MOGOA	0.0091	0.0148	0.0023	0.0017	0.0498
MOPSO	0.0092	0.0055	0.0098	0.0012	0.0165
NSGA-II	0.0827	0.0054	0.0804	0.0773	0.0924
MODA	0.0061	0.0051	0.0038	0.0022	0.0163
MOALO	0.0198	0.0075	0.0196	0.0106	0.0330
ZDT1 with 3 objectives					
MOGOA	0.0114	0.0079	0.0081	0.0068	0.0289
MOPSO	0.0203	0.0013	0.0203	0.0189	0.0225
NSGA-II	0.0626	0.0179	0.0584	0.0371	0.0847
MODA	0.00916	0.0053	0.0063	0.0048	0.0191
MOALO	0.01982	0.0075	0.0196	0.0106	0.0330

the test functions in this test suite are not multi-modal. To benchmark the performance of the proposed algorithm on more challenging test beds, this subsection employs the CEC2009 test functions. These functions are of the most difficult test functions in the literature of multi-objective optimization and able to confirm whether the superiority of MOGOA is significant or not. The mathematical formulation of these test functions are given in the Appendix. The results of MOGOA on CEC2009 test functions are presented in Tables 2, 3, and 4 and compared to MOPSO and MOEA/D.

Table 2 shows that the MOGOA algorithm provides better results on six of the CEC2009 test functions. The test functions are UF3, UF5, UF7, UF8, UF9, and UF10. IGD quantifies the convergence of algorithms, so these results show that the proposed algorithm is able to find very accurate approximations of true Pareto optimal solutions for the given multi-objective problems. High convergence, however, might result in poor coverage. Inspecting the results of Tables 3 and 4, it may be seen that the coverage of the MOGOA algorithm tends to be better than those of MOPSO and MOEA/D on the majority of CEC2009 test functions. This shows that the proposed algorithm benefits from high coverage as well.

The results of the preceding tables were collected and presented considering 30 independent runs. The average, median, standard deviation, maximum, and minimum statistical metrics show how well the proposed algorithm performs on average. To see how significant the superiority of the proposed algorithm is considering each run and prove that the

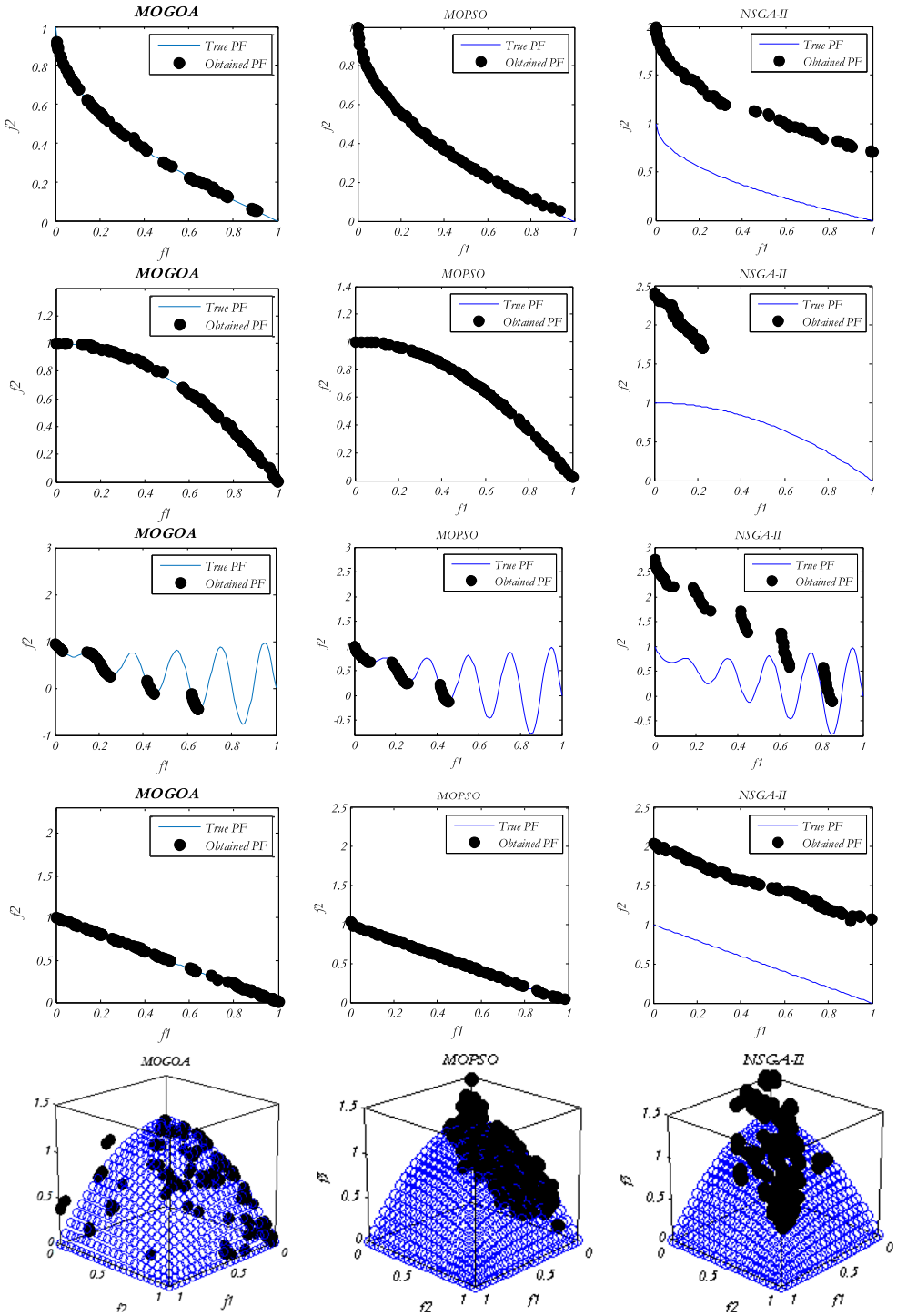

Fig. 5 Best Pareto optimal front obtained by the multi-objective algorithms on test functions

results were not obtained by chance, the Wilcoxon rank-sum test is conducted in this subsection as well. The p-values that are less than 0.05 could be considered as strong evidence against the null hypothesis.

For this statistical test, the best algorithm in each test function is chosen and compared with other algorithms independently. For example, if the best algorithm is MOGOA, the pairwise comparison is done between MOGOA/MOPSO and MOGOA/MOEA/D. The results are presented in Tables 5 and 6. It is evident in these tables that the superiority of the proposed algorithm is statistically significant on the majority of test cases. The MOGOA tends

Table 2 Statistical results for IGD on UF1 to UF10

IGD	MOGOA	MOPSO	MOEA/D	MOGOA	MOPSO	MOEA/D	MOGOA	MOPSO	MOEA/D
	UF1 (bi-objective)			*UF2 (bi-objective)*			*UF3 (bi-objective)*		
Average	0.1811	0.1370	0.1871	0.0959	0.0604	0.1223	0.2380	0.3140	0.2886
Median	0.1892	0.1317	0.1829	0.0894	0.0484	0.1201	0.2166	0.3080	0.2893
STD. Dev.	0.0250	0.0441	0.0507	0.0386	0.0276	0.0107	0.0662	0.0447	0.0159
Worst	0.2100	0.2279	0.2464	0.1681	0.1305	0.1437	0.3690	0.3777	0.3129
Best	0.1430	0.0899	0.1265	0.0488	0.0370	0.1049	0.1682	0.2565	0.2634
	UF4 (bi-objective)			*UF5 (bi-objective)*			*UF6 (bi-objective)*		
Average	0.0702	0.1360	0.0681	1.1559	2.2024	1.2915	0.7771	0.6475	0.6881
Median	0.0696	0.1343	0.0685	1.1470	2.1257	1.3376	0.7345	0.5507	0.6984
STD. Dev.	0.0048	0.0074	0.0021	0.1661	0.5530	0.1349	0.2769	0.2661	0.0553
Worst	0.0788	0.1519	0.0704	1.4174	3.0384	1.4675	1.3288	1.2428	0.7401
Best	0.0639	0.1273	0.0647	0.8978	1.4648	1.1231	0.4939	0.3793	0.5524
	UF7 (bi-objective)			*UF8 (tri-objective)*			*UF9 (tri-objective)*		*UF10 (tri-objective)*
Average	0.1726	0.3540	0.4552	0.2805	0.5367	0.4427	0.4885	0.9043	1.6372
Median	0.1567	0.3873	0.4377	0.2497	0.5364	0.4330	0.4145	0.8928	1.5916
STD. Dev.	0.0633	0.2044	0.1898	0.0749	0.1826	0.0609	0.1445	0.1848	0.2988
Worst	0.3320	0.6151	0.6770	0.4532	0.7964	0.5662	0.7221	1.2285	2.1622
Best	0.1150	0.0540	0.0290	0.2154	0.2453	0.3742	0.3336	0.6432	1.2201

to provide p-values greater than 0.05, which shows that this algorithm is highly competitive on the test cases where it not the best.

To sum up, the results show that MOGOA is very promising and competitive compared to the current well-regarded algorithms. The reasons for this can be summarized in two key features: high convergence and coverage. Superior convergence is due to the target selection, in which one of the best non-dominated solutions always updates the position of others. Another advantage is the high coverage of

Table 3 Statistical results for SP on UF1 to UF10

IGD	MOGOA	MOPSO	MOEA/D	MOGOA	MOPSO	MOEA/D	MOGOA	MOPSO	MOEA/D
	UF1 (bi-objective)			*UF2 (bi-objective)*			*UF3 (bi-objective)*		
Average	0.0012	0.0090	0.0038	0.0007	0.0083	0.0088	0.0019	0.0070	0.0268
Median	0.0012	0.0086	0.0038	0.0001	0.0081	0.0086	0.0006	0.0068	0.0251
STD. Dev.	0.0011	0.0025	0.0015	0.0011	0.0017	0.0008	0.0024	0.0017	0.0206
Worst	0.0031	0.0146	0.0067	0.0031	0.0125	0.0104	0.0055	0.0101	0.0626
Best	0.0000	0.0067	0.0021	0.0000	0.0062	0.0080	0.0000	0.0048	0.0008
	UF4 (bi-objective)			*UF5 (bi-objective)*			*UF6 (bi-objective)*		
Average	0.0001	0.0067	0.0073	0.0007	0.0048	0.0028	0.0003	0.0208	0.0063
Median	0.0000	0.0066	0.0073	0.0007	0.0049	0.0001	0.0002	0.0124	0.0000
STD. Dev.	0.0002	0.0009	0.0006	0.0005	0.0041	0.0055	0.0004	0.0326	0.0127
Worst	0.0006	0.0081	0.0084	0.0014	0.0121	0.0162	0.0011	0.1114	0.0303
Best	0.0000	0.0055	0.0061	0.0001	0.0001	0.0000	0.0000	0.0022	0.0000
	UF7 (bi-objective)			*UF8 (tri-objective)*			*UF9 (tri-objective)*		*UF10 (tri-objective)*
Average	0.0001	0.0067	0.0054	0.0175	0.0268	0.0139	0.0234	0.0067	0.0199
Median	0.0000	0.0066	0.0044	0.0177	0.0264	0.0123	0.0235	0.0067	0.0207
STD. Dev.	0.0001	0.0029	0.0030	0.0085	0.0083	0.0101	0.0041	0.0041	0.0035
Worst	0.0002	0.0124	0.0117	0.0320	0.0447	0.0320	0.0309	0.0123	0.0267
Best	0.0000	0.0033	0.0008	0.0069	0.0153	0.0000	0.0172	0.0000	0.0154

Table 4 Statistical results for MS on UF1 to UF10

IGD	MOGOA	MOPSO	MOEA/D	MOGOA	MOPSO	MOEA/D	MOGOA	MOPSO	MOEA/D
	UF1 (bi-objective)			*UF2 (bi-objective)*			*UF3 (bi-objective)*		
Average	0.7270	0.6454	0.5177	0.8845	0.9121	0.8720	0.6051	0.6103	0.2399
Median	0.7607	0.6632	0.5954	0.8824	0.9164	0.8744	0.6513	0.6161	0.2294
STD. Dev.	0.1507	0.1929	0.1661	0.0353	0.0256	0.0056	0.1100	0.1058	0.1213
Worst	0.4899	0.2659	0.3149	0.8150	0.8665	0.8599	0.4026	0.3817	0.0898
Best	0.9120	0.9523	0.7413	0.9360	0.9530	0.8779	0.7060	0.7715	0.4786
	UF4 (bi-objective)			*UF5 (bi-objective)*			*UF6 (bi-objective)*		
Average	0.9050	0.8128	0.8832	0.2379	0.2793	0.2922	0.2525	0.2744	0.0968
Median	0.9060	0.8132	0.8813	0.2213	0.2865	0.2917	0.2109	0.2292	0.0001
STD. Dev.	0.0139	0.0137	0.0181	0.1131	0.0958	0.0347	0.1294	0.1129	0.2072
Worst	0.8834	0.7944	0.8532	0.1150	0.1557	0.2383	0.0695	0.1544	0.0000
Best	0.9310	0.8345	0.9139	0.4894	0.4383	0.3438	0.4600	0.5252	0.5948
	UF7 (bi-objective)			*UF8 (tri-objective)*			*UF9 (tri-objective)*		*UF10 (tri-objective)*
Average	0.8460	0.4293	0.5632	0.4417	0.5081	0.1938	0.1982	0.3233	0.1302
Median	0.8400	0.2952	0.6327	0.4980	0.5060	0.1700	0.1657	0.2738	0.1091
STD. Dev.	0.0792	0.2755	0.2421	0.1586	0.1614	0.0730	0.1635	0.1237	0.0626
Worst	0.7029	0.1446	0.1496	0.1661	0.2272	0.1175	0.0677	0.1704	0.0649
Best	0.9570	0.8771	0.9915	0.6342	0.7148	0.2940	0.6424	0.5290	0.2540

the MOGOA algorithm, which is because of both archive maintenance mechanism and the selection of target. Since the solutions are always discarded from most populated segments and targets are chosen from the least populated segments of the archive, MOGOA improves the diversity and coverage of solutions across all objectives. Despite these benefits, MOGOA is supposed to be applied to problems with three and maximum four objectives. As a Pareto dominance-based algorithm, MOGOA becomes less effective proportional to the number of objectives. This is due to the fact that in problems with more than four objectives, a large number of solutions are non-dominated, so the archive become full quickly. Therefore, the MOGOA algorithm is suitable for solving problems with less than four objectives. In addition, this algorithm is suitable only for problems with continuous variables and requires legit modifications to be used in problems with discrete variables.

The results proved that MOGOA can be very effective for solving optimization problems with multiple objectives. The MOGOA algorithm showed high convergence and coverage. The superior convergence of MOGOA is due to the updating solutions around the best non-dominated solutions obtained so far. The solutions tend towards the best solutions. Also, the high convergence originates from the adaptive mechanism which accelerates the movements of grasshoppers toward the best non-dominated solutions obtained so far in the repository. The high coverage of MOGOA is because of the repository maintenance and

Table 5 P-values obtained from the ranksum test on UF1 to UF7

Test function	IGD			SP			MS		
	MOGOA	MOPSO	MOEA/D	MOGOA	MOPSO	MOEA/D	MOGOA	MOPSO	MOEA/D
UF1	0.0091	N/A	0.0140	N/A	0.0002	0.0001	N/A	0.3447	0.0001
UF2	0.0113	N/A	0.0017	N/A	0.0002	0.0001	0.0757	N/A	0.0001
UF3	N/A	0.0173	0.0539	N/A	0.0006	0.1153	0.6776	N/A	0.0001
UF4	0.4727	0.0002	N/A	N/A	0.0002	0.0001	N/A	0.0002	0.0001
UF5	N/A	0.0002	0.0640	N/A	0.0376	0.0001	0.2730	N/A	0.1153
UF6	0.2413	N/A	0.2123	N/A	0.0002	0.0001	0.5708	N/A	0.4429
UF7	N/A	0.1212	0.0028	N/A	0.0002	0.0001	N/A	0.0036	0.0001

Table 6 P-values obtained from the ranksum test on UF8 to UF10

Test function	IGD		SP		MS	
	MOGOA	MOPSO	MOGOA	MOPSO	MOGOA	MOPSO
UF8	N/A	0.0022	N/A	0.0376	0.4727	N/A
UF9	N/A	0.9698	N/A	0.0257	0.6232	N/A
UF10	N/A	0.0002	N/A	0.0002	N/A	0.0010

target selection mechanisms. When the repository becomes full, non-dominated solutions in populated regions are discarded by MOGOA, which results in improving the distribution of solutions along the entire front.

The procedure of selecting the target also emphasizes coverage because it selects solutions from the least populated regions to be explored and exploited by the swarm. It is worth mentioning here that since the updating mechanism of the target in MOGOA is identical to those of GOA, MOGOA inherits high exploration, local solutions avoidance, exploitation, and fast convergence rate from this algorithm. Also, the repulsion and comfort zone of this algorithm cause high exploration and consequently discovering new paths towards the undiscovered regions of the true Pareto optimal front. Therefore, the MOGOA algorithm can avoid local fronts and converge towards the true Pareto optimal front.

5 Conclusion

This work proposed a nature-inspired multi-objective algorithm mimicking the interaction of individuals in a swarm of grasshoppers. At first, a mathematical model was employed to simulate the behaviour of grasshopper swarm and proposed a single-objective optimization algorithm. An archive and target selection mechanism was then integrated into this algorithm to solve multi-objective problems. A set of test functions was used to test the performance of the proposed MOGOA algorithm. The results were compared with those of MOPSO and NSGA-II as the best algorithms in the literature. It was observed that the MOGOA algorithm is very efficient and competitive in finding an accurate estimation of Pareto optimal front with high distribution across all objectives. In addition, it was discussed that accurate estimated solutions are due the high convergence of MOGOA, and the good distribution is because of the high exploration. Also, the target selection mechanism and archive maintenance promote exploration and distribution of solutions. For future works, it is recommended to investigate the effectiveness of different constraint handling techniques to solve multi-objective algorithms with constraints using MOGOA.

Appendix: Multi-objective test problems utilised in this work

Table 7 ZDT test suite

ZDT1:

Minimise: $f_1(x) = x_1$

Minimise: $f_2(x) = g(x) \times h(f_1(x), g(x))$

Where: $G(x) = 1 + \frac{9}{N-1}\sum_{i=2}^{N} x_i$

$h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}}$

$0 \le x_i \le 1, 1 \le i \le 30$

ZDT2:

Minimise: $f_1(x) = x_1$

Minimise: $f_2(x) = g(x) \times h(f_1(x), g(x))$

Where: $G(x) = 1 + \frac{9}{N-1}\sum_{i=2}^{N} x_i$

$h(f_1(x), g(x)) = 1 - \left(\frac{f_1(x)}{g(x)}\right)^2$

$0 \le x_i \le 1, 1 \le i \le 30$

ZDT3:

Minimise: $f_1(x) = x_1$

Minimise: $f_2(x) = g(x) \times h(f_1(x), g(x))$

Where: $G(x) = 1 + \frac{9}{29}\sum_{i=2}^{N} x_i$

$h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} - \left(\frac{f_1(x)}{g(x)}\right)\sin(10\pi f_1(x))$

$0 \le x_i \le 1, 1 \le i \le 30$

ZDT1 with linear PF:

Minimise: $f_1(x) = x_1$

Minimise: $f_2(x) = g(x) \times h(f_1(x), g(x))$

Where: $G(x) = 1 + \frac{9}{N-1}\sum_{i=2}^{N} x_i$

$h(f_1(x), g(x)) = 1 - \frac{f_1(x)}{g(x)}$

$0 \le x_i \le 1, 1 \le i \le 30$

ZDT2 with three objectives:

Minimise: $f_1(x) = x_1$

Minimise: $f_2(x) = x_2$

Minimise: $f_3(x) = g(x) \times h(f_1(x), g(x)) \times h(f_2(x), g(x))$

Where: $G(x) = 1 + \frac{9}{N-1}\sum_{i=3}^{N} x_i$

$h(f_1(x), g(x)) = 1 - \left(\frac{f_1(x)}{g(x)}\right)^2$

$0 \le x_i \le 1, 1 \le i \le 30$

Table 8 Bi-objective test problems (CEC2009)

Name	Mathematical formulation
UF1	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left[x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right)\right]^2$, $f_2 = 1-\sqrt{x} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}\left[x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right)\right]^2$ $J_1 = \{j \lvert j\ is\ odd\ and\ 2 \le j \le n\}$, $J_2 = \{j \lvert j\ is\ even\ and\ 2 \le j \le n\}$
UF2	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} y_j^2$, $f_2 = 1-\sqrt{x} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} y_j^2$ $J_1 = \{j \lvert j\ is\ odd\ and\ 2 \le j \le n\}$, $J_2 = \{j \lvert\ j\ is\ even\ and\ 2 \le j \le n\}$ $y_j = \begin{cases} x_j - \left[0.3x_1^2\cos\left(24\pi x_1 + \frac{4j\pi}{n}\right) + 0.6x_1\right]\cos\left(6\pi x_1 + \frac{j\pi}{n}\right) & if j \in J_1 \\ x_j - \left[0.3x_1^2\cos\left(24\pi x_1 + \frac{4j\pi}{n}\right) + 0.6x_1\right]\sin\left(6\pi x_1 + \frac{j\pi}{n}\right) & if j \in J_2 \end{cases}$
UF3	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\left(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right)$ $f_2 = \sqrt{x}_1 + \frac{2}{\lvert J_2\rvert}\left(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_2}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right)$ J_1 and J_2 are the same as those of $UF1$, $y_j = x_j - x_1^{0.5\left(1.0+\frac{3(j-2)}{n-2}\right)}$, $j = 2, 3, \ldots, n$
UF4	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} h\left(y_j\right)$, $f_2 = 1 - x_2 + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} h\left(y_j\right)$ J_1 and J_2 are the same as those of $UF1$, $y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right)$, $j = 2, 3, \ldots, n$, $h(t) = \frac{\lvert t\rvert}{1+e^{2\lvert t\rvert}}$
UF5	$f_1 = x_1 + \left(\frac{1}{2N} + \epsilon\right)\lvert\sin(2N\pi x_1)\rvert + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} h\left(y_i\right)$, $f_1 = 1 - x_1 + (\frac{1}{2N} + \epsilon)\lvert\sin(2N\pi x_1)\rvert + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} h\left(y_i\right)$ J_1 amd J_2 are identical to those of $UF1$, $\epsilon > 0$, $y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right)$, $j = 2, 3, \ldots, n$ $h(t) = 2t^2 - \cos(4\pi t) + 1$
UF6	$f_1 = x_1 + \max\left\{0, 2\left(\frac{1}{2N} + \epsilon\right)\sin(2N\pi x_1)\right\} + \frac{2}{\lvert J_1\rvert}(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 1))$ $f_2 = 1 - x_1 + \max\left\{0, 2\left(\frac{1}{2N} + \epsilon\right)\sin(2N\pi x_1)\right\}\frac{2}{\lvert J_2\rvert}(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 1))$ J_1 amd J_2 are identical to those of $UF1$, $\epsilon > 0$, $y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right)$, $j = 2, 3, \ldots, n$
UF7	$f_1 = \sqrt[5]{x_1} + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} y_j^2$, $f_2 = 1 - \sqrt[5]{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} y_j^2$ J_1 amd J_2 are identical to those of $UF1$, $\epsilon > 0$, $y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right)$, $j = 2, 3, \ldots, n$

Table 9 Tri-objective test problems (CEC2009)

Name	Mathematical formulation
UF8	$f_1 = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left(x_j - 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right)^2\right)$ $f_2 = \cos(0.5x_1\pi)\sin(0.5x_2\pi) + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}\left(x_j - 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right)^2\right)$ $f_3 = \sin(0.5x_1\pi) + \frac{2}{\lvert J_3\rvert}\sum_{j\in J_3}\left(x_j - 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right)^2\right)$ $J_1 = \{j \lvert 3 \le j \le n,\ and\ j-1\ is\ a\ multiplication\ of\ 3\}$, $J_2 = \{j \lvert 3 \le j \le n,\ and\ j-2\ is\ a\ multiplication\ of\ 3\}$, $J_3 = \{j \lvert 3 \le j \le n,\ and\ j\ is\ a\ multiplication\ of\ 3\}$
UF9	$f_1 = 0.5\left[\max\left\{0, (1+\epsilon)\left(1-4(2x_1-1)^2\right)\right\} + 2x_1\right]x_2 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left(x_j - 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right)^2\right)$ $f_2 = 0.5\left[\max\left\{0, (1+\epsilon)\left(1-4(2x_1-1)^2\right)\right\} + 2x_1\right]x_2 + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}\left(x_j - 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right)^2\right)$ $f_3 = 1 - x_2 + \frac{2}{\lvert J_3\rvert}\sum_{j\in J_3}\left(x_j - 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right)^2\right)$ $J_1 = \{j \lvert 3 \le j \le n,\ and\ j-1\ is\ a\ multiplication\ of\ 3\}$, $J_2 = \{j \lvert 3 \le j \le n, and\ j-2\ is\ a\ multiplication\ of\ 3\}$, $J_3 = \{j \lvert 3 \le j \le n, and\ j\ is\ a\ multiplication\ of\ 3\}$, $\epsilon = 0.1$
UF10	$f_1 = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left[4y_j^2 - \cos\left(8\pi y_j\right) + 1\right]$ $f_2 = \cos(0.5x_1\pi)\sin(0.5x_2\pi) + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_1}\left[4y_j^2 - \cos\left(8\pi y_j\right) + 1\right]$ $f_3 = \sin(0.5x_1\pi) + \frac{2}{\lvert J_3\rvert}\sum_{j\in J_1}\left[4y_j^2 - \cos\left(8\pi y_j\right) + 1\right]$ $J_1 = \{j \lvert 3 \le j \le n,\ and\ j-1\ is\ a\ multiplication\ of\ 3\}$, $J_2 = \{j \lvert 3 \le j \le n,\ and\ j-2\ is\ a\ multiplication\ of\ 3\}$, $J_3 = \{j \lvert 3 \le j \le n,\ and\ j\ is\ a\ multiplication\ of\ 3\}$

References

1. Holland JH, Reitman JS (1977) Cognitive systems based on adaptive algorithms. ACM SIGART Bulletin, pp 49–49
2. Tsai C-F, Eberle W, Chu C-Y (2013) Genetic algorithms in feature and instance selection. Knowl-Based Syst 39:240–247
3. Lin M-H, Tsai J-F, Yu C-S (2012) A review of deterministic optimization methods in engineering and management. Mathematical Problems in Engineering, vol 2012
4. Shmoys DB, Swamy C (2004) Stochastic optimization is (almost) as easy as deterministic optimization. In: Proceedings of the 45th annual IEEE symposium on foundations of computer science, 2004, pp 228–237
5. Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization. IEEE Comput Intell Mag 1:28–39
6. Kennedy J (2011) Particle swarm optimization. In: Encyclopedia of machine learning, edn. Springer, pp 760–766
7. Knowles J, Corne D (1999) The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In: Proceedings of the 1999 congress on evolutionary computation, 1999. CEC 99
8. Storn R, Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359
9. Padhye N, Mittal P, Deb K (2013) Differential evolution: performances and analyses. In: 2013 IEEE congress on evolutionary computation (CEC), pp 1960–1967
10. Padhye N, Bhardawaj P, Deb K (2010) Improving differential evolution by altering steps in EC. In: Asia-Pacific conference on simulated evolution and learning, pp 146–155
11. BoussaïD I, Lepagnot J, Siarry P (2013) A survey on optimization metaheuristics. Inf Sci 237:82–117
12. Helbig M, Engelbrecht AP (2013) Performance measures for dynamic multi-objective optimisation algorithms. Inf Sci 250:61–81
13. Padhye N, Zuo L, Mohan CK, Varshney PK (2009) Dynamic and evolutionary multi-objective optimization for sensor selection In sensor networks for target tracking. In: Proceedings of the international joint conference on computational intelligence - volume 1: ICEC, (IJCCI 2009). ScitePress, INSTICC, pp 160–167. doi:10.5220/0002324901600167, ISBN: 978-989-674-014-6
14. Padhye N, Zuo L, Mohan CK, Varshney P (2009) Dynamic and evolutionary multi-objective optimization for sensor selection in sensor networks for target tracking. In: IJCCI, pp 160–167
15. Beyer H-G, Sendhoff B (2007) Robust optimization–a comprehensive survey. Comput Methods Appl Mech Eng 196:3190–3218
16. Coello CAC (1999) A comprehensive survey of evolutionary-based multiobjective optimization techniques. Knowl Inf Syst 1:269–308
17. Deb K (2001) Multi-objective optimization using evolutionary algorithms, vol 16. Wiley
18. von Lücken C., Barán B., Brizuela C (2014) A survey on multi-objective evolutionary algorithms for many-objective problems. Comput Optim Appl 58:707–756
19. Nguyen TT, Yang S, Branke J (2012) Evolutionary dynamic optimization: a survey of the state of the art. Swarm Evol Comput 6:1–24
20. Padhye N, Mittal P, Deb K (2015) Feasibility preserving constraint-handling strategies for real parameter evolutionary optimization. Comput Optim Appl 62:851–890
21. Padhye N, Deb K, Mittal P (2013) An efficient and exclusively-feasible constrained handling strategy for evolutionary algorithms. Technical Report
22. Asrari A, Lotfifard S, Payam MS (2016) Pareto dominance-based multiobjective optimization method for distribution network reconfiguration. IEEE Trans Smart Grid 7:1401–1410
23. Zhou A, Qu B-Y, Li H, Zhao S-Z, Suganthan PN, Zhang Q (2011) Multiobjective evolutionary algorithms: a survey of the state of the art. Swarm Evol Comput 1(3//):32–49
24. Deb K (2014) Multi-objective optimization. In: Search methodologies, edn. Springer, pp 403–449
25. Padhye N, Deb K (2010) Evolutionary multi-objective optimization and decision making for selective laser sintering. In: Proceedings of the 12th annual conference on genetic and evolutionary computation, pp 1259–1266
26. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6:182–197
27. Coello CC, Lechuga MS (2002) MOPSO: A proposal for multiple objective particle swarm optimization. In: Proceedings of the 2002 congress on evolutionary computation, 2002. CEC'02, pp 1051–1056
28. Padhye N (2008) Topology optimization of compliant mechanism using multi-objective particle swarm optimization. In: Proceedings of the 10th annual conference companion on genetic and evolutionary computation, pp 1831–1834
29. Padhye N (2009) Comparison of archiving methods in multi-objective particle swarm optimization (MOPSO): empirical study. In: Proceedings of the 11th annual conference on Genetic and evolutionary computation, pp 1755–1756
30. Alaya I, Solnon C, Ghedira K (2007) Ant colony optimization for multi-objective optimization problems. In: ICTAI (1), pp 450–457
31. Xue F, Sanderson AC, Graves RJ (2003) Pareto-based multi-objective differential evolution. In: The 2003 congress on evolutionary computation, 2003. CEC'03, pp 862–869
32. Knowles JD, Corne DW (2000) Approximating the nondominated front using the Pareto archived evolution strategy. Evol Comput 8:149–172
33. Wolpert D (1997) No free lunch theorem for optimization. In: IEEE transactions on evolutionary computation, pp 467–482
34. Ishibuchi H, Tsukamoto N, Nojima Y (2008) Evolutionary many-objective optimization: a short review. In: IEEE congress on evolutionary computation, pp 2419–2426
35. Marler RT, Arora JS (2004) Survey of multi-objective optimization methods for engineering. Struct Multidiscip Optim 26:369–395
36. Deb K, Padhye N, Neema G (2007) Multiobjective evolutionary optimization-interplanetary trajectory optimization with swing-bys using evolutionary multi-objective optimization. Lect Notes Comput Sci 4683:26–35
37. Jin Y, Olhofer M, Sendhoff B (2001) Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how?
38. Branke J, Deb K, Dierolf H, Osswald M (2004) Finding knees in multi-objective optimization. In: International conference on parallel problem solving from nature, pp 722–731
39. Kollat JB, Reed P (2007) A framework for visually interactive decision-making and design using evolutionary multi-objective optimization (VIDEO). Environ Model Softw 22:1691–1704
40. Topaz CM, Bernoff AJ, Logan S, Toolson W (2008) A model for rolling swarms of locusts. Eur Phys J Special Topics 157:93–109
41. Saremi S, Mirjalili S, Lewis A (2017) Grasshopper optimisation algorithm: theory and application. Adv Eng Softw 105:30–47
42. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6:182–197
43. Zitzler E, Laumanns M, Thiele L, Zitzler E, Zitzler E, Thiele L et al (2001) SPEA2: Improving the strength Pareto evolutionary

algorithm, ed: Eidgenössische Technische Hochschule Zürich (ETH). In: Institut für Technische Informatik und Kommunikationsnetze (TIK)
44. Srinivas N, Deb K (1994) Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evol Comput 2:221–248
45. Zitzler E, Thiele L (1998) Multiobjective optimization using evolutionary algorithms—a comparative case study. In: Parallel problem solving from nature—PPSN V, pp 292–301
46. Deb K, Padhye N (2014) Enhancing performance of particle swarm optimization through an algorithmic link with genetic algorithms. Comput Optim Appl 57:761–794
47. Padhye N, Bhardawaj P, Deb K (2013) Improving differential evolution through a unified approach. J Glob Optim 55:771
48. Zitzler E, Deb K, Thiele L (2000) Comparison of multiobjective evolutionary algorithms: Empirical results. Evol Comput 8:173–195
49. Deb K, Thiele L, Laumanns M, Zitzler E (2002) Scalable multi-objective optimization test problems. In: Proceedings of the 2002 congress on evolutionary computation, 2002. CEC'02, pp 825–830
50. Zhang Q, Zhou A, Zhao S, Suganthan PN, Liu W, Tiwari S (2008) Multiobjective optimization test instances for the CEC 2009 special session and competition. University of Essex, Colchester, UK and Nanyang Technological University, Singapore, Special Session on Performance Assessment of Multi-Objective Optimization Algorithms, Technical Report
51. Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Applic 27:1053–1073
52. Mirjalili S, Jangir P, Saremi S (2016) Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems. Applied Intelligence, pp 1–17