Appl Intell (2018) 48:2268–2283
https://doi.org/10.1007/s10489-017-1074-1

MOGOA algorithm for constrained and unconstrained multi-objective optimization problems

Alaa Tharwat[1,2,5] · Essam H. Houssein[3,5] · Mohammed M. Ahmed[3,5] · Aboul Ella Hassanien[4,5] · Thomas Gabel[1]

Published online: 4 November 2017

Abstract Grasshopper Optimization Algorithm (GOA) was modified in this paper, to optimize multi-objective problems, and the modified version is called Multi-Objective Grasshopper Optimization Algorithm (MOGOA). An external archive is integrated with the GOA for saving the Pareto optimal solutions. The archive is then employed for defining the social behavior of the GOA in the multi-objective search space. To evaluate and verify the effectiveness of the MOGOA, a set of standard unconstrained and constrained test functions are used. Moreover, the proposed algorithm was compared with three well-known optimization algorithms: Multi-Objective Particle Swarm Optimization (MOPSO), Multi-Objective Ant Lion Optimizer (MOALO), and Non-dominated Sorting Genetic Algorithm version 2 (NSGA-II); and the obtained results show that the MOGOA algorithm is able to provide competitive results and outperform other algorithms.

✉ Alaa Tharwat
engalaatharwat@hotmail.com

1 Faculty of Computer Science and Engineering, Frankfurt University of Applied Sciences, 60318 Frankfurt am Main, Germany

2 Faculty of Engineering, Suez Canal University, Ismailia, Egypt

3 Faculty of Computers and Information, Minia University, Minia, Egypt

4 Faculty of Computers and Information, Cairo University, Giza, Egypt

5 Scientific Research Group in Egypt (SRGE), Cairo, Egypt
http://www.egyptscience.net

Keywords Multi-objective optimization · Grasshopper optimization algorithm · Pareto optimal solutions · Evolutionary algorithm · Constrained optimization · Unconstrained optimization

1 Introduction

Solving muli-objective problems is one of the big challenges in different applications such as power and energy [1], Robotics [2–4], bioinformatics [5], mechanical engineering [6], and chemoinformatics [7, 8]. In the multi-objective problem, there is no single optimal solution, but rather a set of alternative solutions represent the optimal solutions. These solutions are optimal when no other solutions in the search space are superior to them when all objectives are considered; these solutions are known as Pareto Optimal (PO) solutions [9]. This problem can be handled by combining the objectives into one single objective with a set of weights. These weights represent the importance of each objective. However, the distribution of the weights does not guarantee finding the optimal solution [10]. A multi-objective formulation of the multi-objective problems is capable of exploring the behavior of the problems across a range of parameters and operating conditions [11]. In the multi-objective optimization algorithm, there is no single objective and the goal is to optimize different objectives. Hence, a set of solutions including the optimal solutions represents various trade-offs between different objectives [12–14].

Evolutionary algorithms were introduced by David Schaffer for optimizing multi-objective problems [15–17]. Since then, a significant number of studies have been introduced for developing multi-objective evolutionary algorithms. The evolutionary algorithms are gradient-free and

they can escape from local optima traps. The literature shows that the evolutionary algorithms can approximate the true Pareto optimal solutions of multi-objective problems effectively. Many evolutionary algorithms were employed for optimizing multi-objective problems such as Non-dominated Sorting Genetic Algorithm (NSGA-I) [18], Non-dominated Sorting Genetic Algorithm version 2 (NSGA-II) [19], Strength-Pareto Evolutionary Algorithm (SPEA) [20], Multi-Objective Particle Swarm Optimization (MOPSO) [21–23], Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) [24], and Pareto-frontier Differential Evolution (PDE) [25]. According to the No Free Lunch (NFL) theorem [26], the superior performance of an optimization algorithm on a class of problems or applications cannot guarantee the similar performance on other problems.

Grasshopper Optimization Algorithm (GOA) was proposed recently and it has been proven to benefit from its high exploration while showing very fast convergence speed toward the optimal solutions [27]. In GOA, the special adaptive mechanism smoothly balances the exploration and exploitation phases which makes the GOA potentially able to cope with the difficulties of multi-objective problems and outperform other optimization algorithms. As reported in [27], the computational complexity of GOA is better than the complexity of many other optimization algorithms. These reasons motivated our attempts to propose a multi-objective optimizer inspired from the GOA. In this paper, we introduce a multi-objective variant of GOA, termed MOGOA, and employed it for optimizing constrained and unconstrained testing functions. A distinguishing feature of MOGOA is that an external archive which was integrated to the GOA algorithm to store non-dominated solutions. Moreover, the multi-objective version of the GOA has been proposed using the features of original GOA.

The rest of the paper is organized as follows. Section 2 presents some of the related work. Definitions and preliminaries of optimization algorithm, i.e., GOA, in a multi-objective search space were introduced in Section 3. In Section 4, the proposed algorithm MOGOA was introduced. Experimental results and discussions are presented in Section 5. Concluding remarks and future work are provided in Section 6.

2 Related work

The main goal of multi-objective optimization algorithms is to search for an accurate approximation of the exact or true Pareto optimal solutions. This problem was solved by aggregating the objectives into a single objective by using weight factors as follows, $F = \sum_{i=1}^{N} w_n f_n$, where N indicates the number of objective functions, w_n is the weighting factors, and f_n represent the objective functions [28, 29]. In single-objective problems, the problem has one point as the global optimal solution; therefore, the solutions can be compared easily [30]. However, finding a set of solutions can be solved by using a wide variety of weight factors which is extremely time-consuming [31]. Moreover, the distribution of the weights does not guarantee finding Pareto optimal solutions [32].

There are some studies that tried to improve this method. For example, two dynamic weighted aggregations were used to adjust the weights over time [33]. However, the main problems were not completely solved using this method. Moreover, it needs to run for several times to approximate the whole Pareto solutions. Deb reported that the metaheuristics were used with the multi-objective optimization problem to overcome some difficulties such as local fronts, infeasible area, isolation of optimum, and diversity of solutions [34].

Recently, multi-objective optimization algorithms were used to approximate the whole Pareto optimal front in a single run. These algorithms address the conflicting objectives and allow the exploration of the behavior of problems across the operating conditions and a range of design parameters. Most of the well-known meta-heuristic optimization such as Genetic Algorithm (GA) [34] and Particle Swarm Optimization (PSO) [12] were used for solving the multi-objective problems.

A modified version of the GA was called Non-dominated Sorting GA (NSGA-II) and it was used for solving multi-objective problems [35]. This algorithm was proposed to solve the problems of the first version NSGA-I [18] such as the high computational cost of non-dominated sorting and lack of considering elitism. In NSGA-II, the solutions are grouped based on the non-dominating sorting method and the fitness for the individuals are defined based on its non-domination level. MOPSO was proposed by Coello et al. [12]. The MOPSO is inspired from PSO [36] and it starts by placing all particles randomly in the problem space. These particles' positions were updated using their own non-dominated solutions, i.e., previous best positions, and the non-dominated solution the swarm has obtained so far, i.e., global best position. The MOPSO is terminated by satisfying of a stopping criterion. MOPSO has a fast convergence speed which makes it prone to parameter termination without finding the Pareto optimal front [37]. In MOPSO, an external archive was utilized for storing and retrieving the obtained Pareto optimal solutions. This external archive was also introduced in [38] to design the adaptive grid in Pareto Archived Evolution Strategy (PAES). In PAES, the archive controller component was used for deciding if a solution should be added to the archive or not. In other words, if a new solution is not dominated by the archive members, it should be added as a new member to the archive.

Moreover, in PAES, the grid component is responsible for making the archive solutions diversified.

Recently, many multi-objective optimization algorithms have been proposed. For example, the Multi-Objective Cat Swarm Optimization (MOCSO) algorithm was proposed in [39] and the results were compared with MOPSO and NSGA-II. Shi Xiangui and Kong Dekui introduced the Multi-Objective Ant Colony Optimization (MOACO) algorithm [40]. Multi-objective Artificial Bee Colony (MOABC) algorithm was proposed in [41], and the proposed algorithm was used for feature selection using fuzzy mutual information. In another research, Multi-objective Gravitational Search Algorithm (MOGSA) was introduced and it outperformed the NSGA-II and MOACO algorithms [42]. Multi Objective Differential Evolution algorithm was proposed and it was compared with NSGA-II and PAES algorithms [43]. In addition, Multi-Objective Cuckoo Search Optimization (MOCSO) was proposed in [44] and Multi-Objective Gray-Wolf Optimization (MOGWO) was introduced in [45]. Further, Multi-objective Teaching Learning-Based Optimization (MOTLO) algorithm was applied for scheduling in turning processes for minimizing makespan and carbon footprint [46]. The recent studies show the ability of the metaheuristic optimization algorithms in handling multi-objective problems. However, all mentioned algorithms are not able to solve all optimization problems according to the NFL theorem. Hence, it is very likely that a new optimization algorithm solves a problem that cannot be solved by one of the existing techniques in the literature. In the next section, a novel multi-objective version of GOA is introduced as an alternative to the current optimization algorithms in the literature for solving multi-objective optimization problems.

3 Preliminaries

3.1 Multi-objective optimization

In Multi-objective Optimization Problem (MOP), the problem has two or more objective functions and it can be formulated as follows:

$$\begin{aligned} \text{Minimize} \quad & F(x) = [f_1(x), f_2(x), \ldots, f_m(x)]^T, \\ \text{Subject to} \quad & : g_i(x) \geq 0, \ i = 1, 2, \ldots, n, \\ & \quad h_i(x) = 0, \ i = 1, 2, \ldots, o, \\ & \quad L_i \leq x_i \leq U_i, \ i = 1, 2, \ldots, p, \end{aligned} \tag{1}$$

where $x = [x_1, x_2, \ldots, x_d]$ represents a vector of design variables, d represents the number of variables, m is the number of objective functions, g_i represents the i^{th} inequality constraint, n is the number of inequality constraints, $h_i(x)$ is the i^{th} equality constraint, o is the number of equality constraints, and $[L_i, U_i]$ are the boundaries for the variable x_i. In MOP, the problem can be solved by storing a set of best solutions in an external archive ($\mathcal{A}$). This archive stores a historical record that is created for the non-dominated solutions found along the search phases. In the initialization phase, the archive is initialized and it is updated iteratively, and the best solutions are defined as non-dominated solutions or Pareto optimal solutions [47]. In multi-objective problems, the solutions cannot be compared using relational operators. This is due to multi-criterion comparison metrics. In this case, a solution can be considered as a non-dominated solution if it satisfies the following conditions:

1. **Pareto dominance**: Given two vectors $W = (w_1, w_2, \ldots, w_n)$ and $V = (v_1, v_2, \ldots, v_n)$. W dominates V if and only if W is partially less than V in the objective space as follows:

$$\begin{cases} f_i(W) \leq f_i(V) \ \forall \ i \ , \ i = 1, 2, \ldots, m, \\ f_i(W) < f_i(V) \ \exists \ i, \end{cases} \tag{2}$$

   where m represents the number of objective functions [48].
2. **Pareto optimal solution**: vector W represents a Pareto optimal solution if and only if any other obtained solutions cannot dominate W.

A set of Pareto optimal solutions is called a Pareto (Optimal Front) $PF_{Optimal}$ and it consists of a set of non-dominated solutions. The goal for any optimization algorithm is finding the most accurate approximation of true Pareto optimal solutions, i.e., convergence, with uniform distributions across all objectives [49].

Figure 1 shows an overview of non-dominated solutions in MOP. As shown, the problem is multi-objective and it has two objective functions, i.e. the space is two-dimensional, and the goal is to find a solution that minimizes the objective functions. In the figure, the values for f_1 and f_2 of the C solution are higher than the A and B solutions. Hence, C is dominated by A and B. On the other hand, the A and B solutions are considered as non-dominated solutions because neither of them dominates the other.

3.1.1 Performance metric parameters

For fair comparisons among different types of multi-objective evolutionary algorithms, in this study, three well-known assessment measures are used to evaluate the performance of metaheuristic algorithms. The details of each measure are explained below.

Metric of spacing The goal of this metric is to show the distribution of non-dominated solutions which are obtained

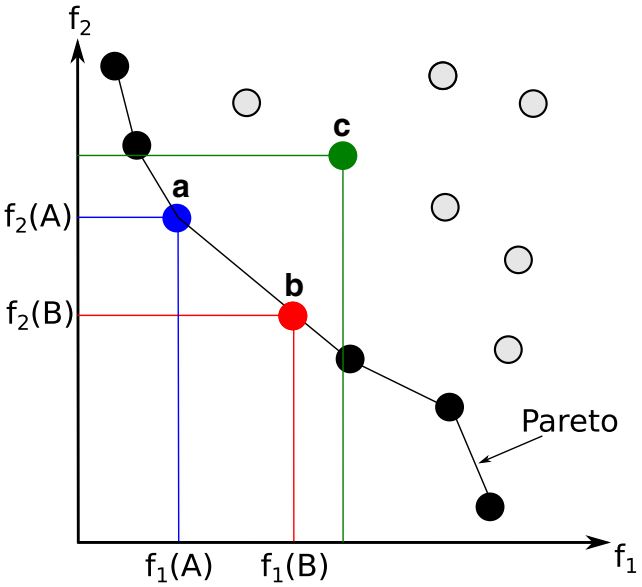

Fig. 1 Illustrative example of the optimal Pareto solution in two-dimensional space, i.e., two objective functions. The solution C is dominated by the A and B solutions

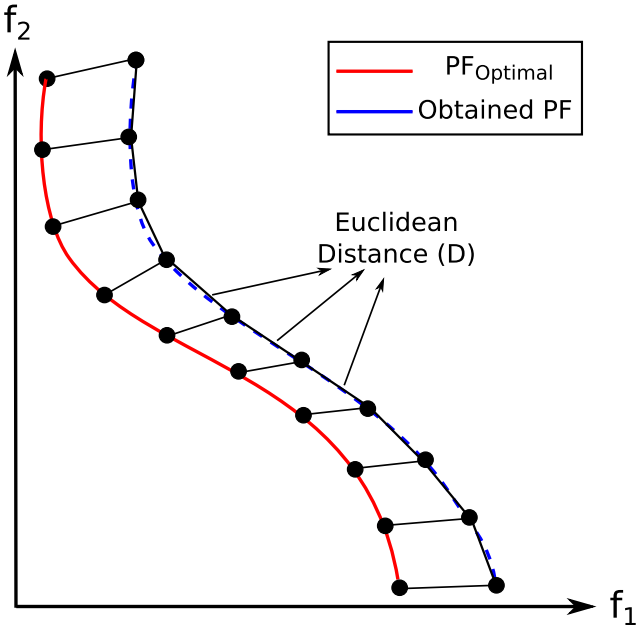

Fig. 2 Illustration of the metric of spacing for MOPs

by a specific algorithm [34]. This metric is defined as follows:

$$S = \sqrt{\frac{1}{n_{pf} - 1} \sum_{i=1}^{n_{pf}} (d_i - \hat{d})^2} \tag{3}$$

where S is the metric of spacing, d_i represents the Euclidean distance between the i^{th} member in PF_g and nearest member in PF_g, PF_g is the generated Pareto front (see Fig. 2), and $\hat{d}$ is the average of all distances. The Euclidean distance is defined in (4). A small value of S gives the best uniform distribution in PF_g, and the value of S will be zero when $d_i = \hat{d}$, i.e., all non-dominated solutions are uniformly distributed.

$$d(a, b) = d(b, a) = \sqrt{\sum_{i=1}^{n} (f_{ia} - f_{ib})^2} \tag{4}$$

where $a = (f_{1a}, f_{2a}, f_{3a}, \ldots, f_{na})$ and $b = (f_{1b}, f_{2b}, f_{3b}, \ldots, f_{nb})$ represents two points on the PF_g.

Metric of spread The spread metric (Δ) was proposed by Deb, and it determines the extent of spread attained by the non-dominated solutions which are obtained from a specified algorithm [34]. Hence, this metric can analyze how the obtained solution is extended across the Pareto optimal fronts, and it is defined as follows:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{n_{pf}} |d_i - \hat{d}|}{d_f + d_l + (n_{pf} - 1)\hat{d}} \tag{5}$$

where d_f and d_l represent the Euclidean distances between the extreme solutions in $PF_{Optimal}$ and PF_g, respectively, as shown in Fig. 3, d_i indicates the Euclidean distance between each point in PF_g and the closest point in $PF_{Optimal}$, n_{pf} is the total number of members in PF_g, and $\hat{d}$ is the average of all distances. As indicated in (5), the value of Δ is always greater than or equal to zero, and a small value of Δ indicates better spread of the obtained solution; $\Delta = 0$ is the best solution indicating that extreme solutions of $PF_{Optimal}$ have been found and $d_i = \hat{d}$ for all non-dominated points.

Generational distance metric The generational distance (GD) metric was first introduced by Veldhuizen and Lamont, and the goal of this metric is to show the capability of different problems which are used for finding a set of non-dominated solutions having the lowest distance with the $PF_{Optimal}$ [50]. Therefore, the algorithm with the minimum GD results has the best convergence to $PF_{Optimal}$ [51]. The definition of GD is as follows:

$$GD = \frac{1}{n_{pf}} \sqrt{\sum_{i=1}^{n_{pf}} d_i^2} \tag{6}$$

where n_{pf} represents the number of members in the generated Pareto front or the obtained Pareto front PF_g and d_i is the Euclidean distance between i^{th} member in PF_g and the nearest member in $PF_{Optimal}$.

Figure 4 displays an illustration of the GD metric in two-dimensional space. In GD metric, the best obtained value is

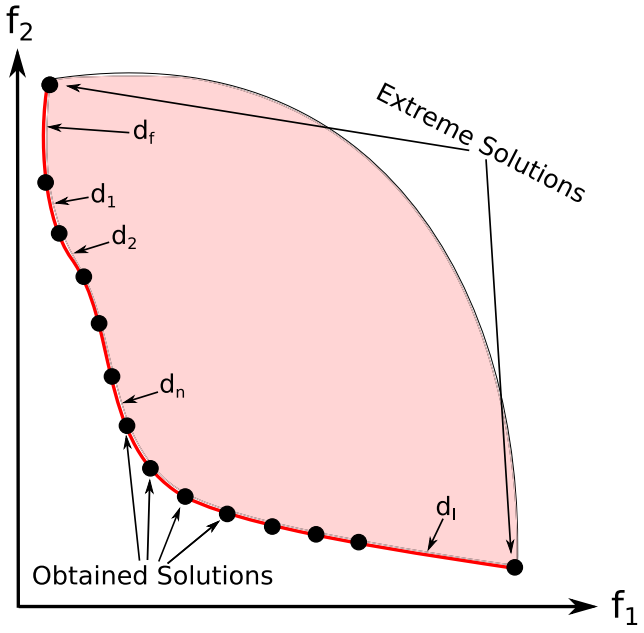

Fig. 3 Illustration of the spread metric (Δ) for MOPs

equal to zero which corresponds to PF_g exactly covers the $PF_{Optimal}$ [52].

3.2 Grasshopper optimization algorithm (GOA)

Grasshopper Optimization Algorithm (GOA) is a new nature-inspired algorithm that was proposed by Mirjalili et al. [27]. This algorithm as many optimization algorithms has two phases: exploration and exploitation. In the exploration phase, the search agents are encouraged to move for searching or exploring different regions in the search space. While in the exploitation phase, the agents move locally to enhance the current solutions. These two phases can be implemented using GOA.

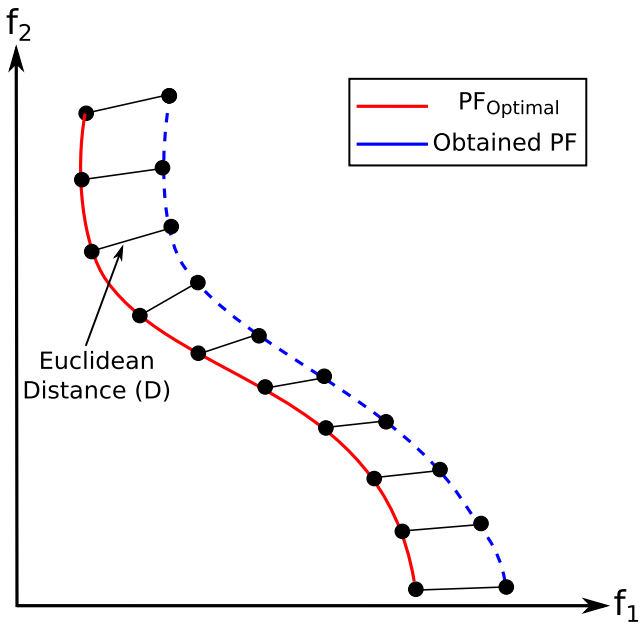

Fig. 4 Illustration of the GD metric for MOPs

Mathematically, the GOA simulates the swarming behavior as follows:

$$X_i = S_i + G_i + A_i \tag{7}$$

where X_i is the position for the i^{th} grasshopper, S_i is the social interaction for the i^{th} grasshopper, G_i represents the gravity force for the i^{th} grasshopper, and A_i represents the wind advection for the i^{th} grasshopper. The random behavior of the GOA can be written as follows, $X_i = r_1 S_i + r_2 G_i + r_3 A_i$, where r_1, r_2, and r_3 are random numbers in [0, 1].

The social interaction is given by:

$$S_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} s(d_{ij})\hat{d_{ij}} \tag{8}$$

where N is the total number of grasshoppers, d_{ij} refers to the distance between the i^{th} and the j^{th} grasshoppers as follows, $d_{ij} = |x_j - x_i|$, s is a function that defines the strength of social forces and it is defined as in (9), and $\hat{d_{ij}}$ represents a unit vector from the i^{th} grasshopper to the j^{th} grasshopper as follows, $\hat{d_{ij}} = \frac{x_j - x_i}{d_{ij}}$.

$$s(r) = f e^{\frac{-r}{l}} - e^{-r} \tag{9}$$

where f is the intensity of attraction and l represents the attractive length scale. GOA has three different zones, namely, comfort, attraction, and repulsion zones; and the values of f and l change the comfort zone or comfortable distance which results in different social behaviors in artificial grasshoppers (see Fig. 5). It is worth mentioning that the attraction or repulsion regions are very small for some values, e.g., $l = 1.0$ and $f = 1.0$. In this study, we have chosen $l = 1.5$ and $f = 0.5$ as in the original paper of GOA [27]. Moreover, the function s divides the space between two grasshoppers into attraction region, comfort zone, and repulsion region.

The G_i component in (7) is defined as follows:

$$G_i = -g\hat{e_g} \tag{10}$$

where g indicates the gravitational constant and $\hat{e_g}$ is a unit vector toward the center of earth.

The third component in (7), i.e., A_i, is defined as follows:

$$A_i = u\hat{e_w} \tag{11}$$

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

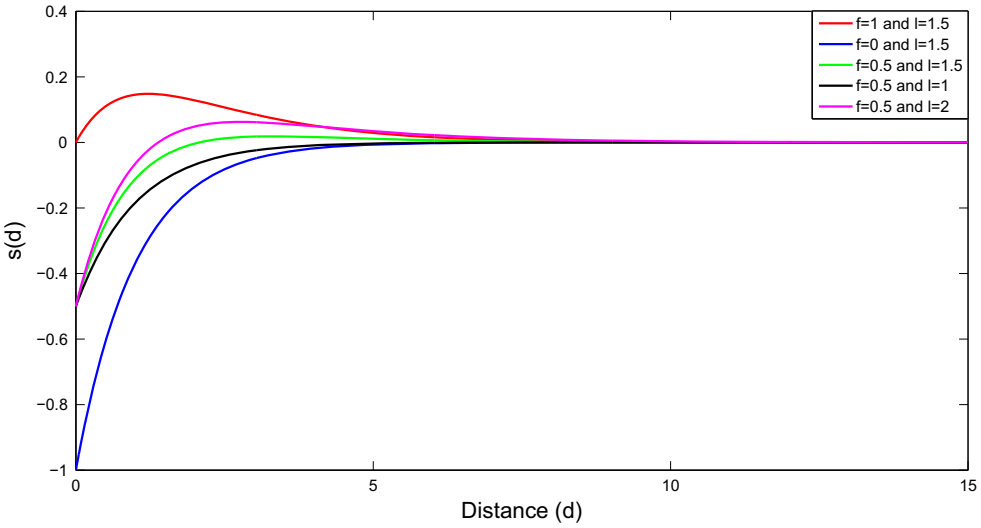

Fig. 5 Behavior of the function s with different values of l and f

Substituting the S, G, and A components in (7) as follows:

$$X_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} s(|x_j - x_i|)\frac{x_j - x_i}{d_{ij}} - g\hat{e_g} + u\hat{e_w} \tag{12}$$

As reported in [27], (12) was not used in swarm simulation and optimization. This is because the grasshoppers quickly reach the comfort zone; hence, the swarm does not converge to the optimal solution. Therefore, (12) prevents the GOA from exploring and exploiting the search space around the current solutions. Some modifications were added to (12) to solve this problem as follows [27]:

$$X_i^d = c\left(\sum_{\substack{j=1 \\ j \neq i}}^{N} c\frac{ub_d - lb_d}{2}s(|x_j^d - x_i^d|)\frac{x_j - x_i}{d_{ij}}\right) + \hat{T}_d \tag{13}$$

where ub_d and lb_d represent the upper and lower bounds, respectively, in the d^{th} dimension, $\hat{T}_d$ indicates the target value in the d^{th} dimension and it also represents the best solution found, c represents a decreasing coefficient and this parameter is used to shrink the attraction zone, comfort zone, and repulsion zone. In (13), S is similar to S component in (12) but the G and A components are not found and the wind direction is always toward the target $\hat{T}_d$.

The parameter c has been used twice in (13) for the following reasons:

- The first c from the left is similar to inertia weight w in PSO [36], loudness in Bat algorithm [53], or $\vec{a}$ in Grey Wolf Optimization [54]. The goal of c is to reduce the movements of the grasshoppers around the target. Thus, it balances between exploration and exploitation of the GOA. The value of c is given by:

$$c = c_{max} - t\frac{c_{max} - c_{min}}{t_{max}} \tag{14}$$

  where c_{max} is the maximum value of c, c_{min} represents the minimum value of c, t is the current iteration, and t_{max} is the maximum number of iterations. In this study, the values of c_{min} and c_{max} were 0.00001 and 1, respectively. Hence, $c \to c_{min}$, where $t \to \infty$.
- The goal of the second c is to decrease the comfort zone, repulsion zone, and attraction zone between grasshoppers. Hence, using this term (c), the repulsion/attraction forces between different grasshoppers decrease proportionally with the number of iterations.

Generally, in GOA, the search agents moved based on their current positions, global best, and the position of all other search agents. While in one of the well-known optimization algorithms PSO, the positions are updated based on the current position, personal best, and global best. Hence, in PSO, none of the other particles contribute to modifying the position of a particle; on the other hand, in GOA, all search agents are required to get involved in finding next position of each search agent. In other words, GOA increases the social capabilities of its agents which reflects how the GOA is more social than PSO, and this will help the GOA to escape from local minima traps.

4 Multi-objective grasshopper optimization algorithm (MOGOA)

Our proposed multi-objective optimization algorithm has two main goals. First, an accurate approximation for the

true Pareto optimal solutions should be obtained. Second, the obtained solution should be well-distributed across all objectives.

Comparing different solutions in multi-objective algorithms cannot be achieved using regular relational operators, instead, a Pareto optimal dominance is utilized. The best Pareto optimal solutions are saved in an external archive. The main difference between the MOGOA and GOA is the process of updating the target which guides the search agents toward promising regions of the problem space. This target can be easily chosen in a single-objective problems by selecting the best solution. In MOGOA, the Pareto optimal solutions are added to the archive and the target is chosen from these solutions to improve the distribution of the current solutions in the archive. This can be achieved through calculating the distance between each solution and a number of neighboring solutions. Next, the number of neighboring solutions is counted and it is used to measure the crowdedness of regions in the Pareto optimal front. Equation (15) defines the probability of choosing the target (P_i) from the current solutions in the archive.

$$P_i = \frac{1}{N_i}, \tag{15}$$

where N_i represents the number of solutions in the neighborhood of the i^{th} solution. Based on P_i, a roulette wheel is used to select the target from the archive. This improves the distribution of less distributed regions of the search space.

The archive has a limitation, i.e., maximum number of solutions, that can be stored in the archive. Increasing the size of the archive increases the computational cost. On the other hand, decreasing the size of the archive may lead to the issue of a full archive. Hence, to solve this problem, solutions in a crowded neighborhood are removed. This will give a chance for a new solution to be stored in less populated regions.

The content of the archive should be updated regularly. This can be achieved by comparing the solution in the archive with a new external solution. In MOGOA, there are two cases:

- when the external solution is dominated by at least one of the archive solutions, the external solution should be thrown away.
- when the external solution is non-dominated with respect to all solutions inside the archive. Thus, a non-dominated solution, i.e., external solution, should be added to the archive. However, if the external solution dominates a solution inside the archive, it should be replaced with it.

Generally, the MOGOA is capable of finding the Pareto optimal solutions, save them in the archive, and their distribution are improved. The steps of the proposed MOGOA algorithm are summarized in Algorithm 1.

Algorithm 1 Pseudo code of the MOGOA

Initialize the grasshopper population $X_i (i = 1, 2, ..., n)$, c_{max}, c_{min}, t_{max} (maximum number of iterations).
Calculate the fitness function for each search agent.
Find the non-dominated PO solutions and initialize external archive ($\mathcal{A}$) with them.
while ($t <= Max_{iter}$) **do**
    Update c using (14).
    for all agents **do**
        Normalize distances between grasshoppers.
        Update grasshoppers' position.
        Calculate the fitness function for each agent.
        Find the non-dominated solutions.
        Update A.
        if ($\mathcal{A}$ is full) **then**
            Run the grid mechanism to remove one of the current archive members.
            Add the new solution to $\mathcal{A}$.
        end if
    end for
    $t = t + 1$
end while
Return $\mathcal{A}$ (contains PO)

It is worth mentioning that the computational complexity of the MOGOA algorithm is $O(MN^2)$, where M and N represent the number of objectives and the number of solutions, respectively, while the computational complexity of NSGA [19] and SPEA [20] algorithms are $O(MN^3)$. This reflects how the proposed MOGOA is faster for finding the optimal or near optimal solutions than some of the state-of-the-art algorithms.

5 Experimental results and discussion

In this section, we describe the results we obtained from a set of experiments for evaluating the proposed MOGOA algorithm. The aim of the first set of experiments is to test our algorithm using unconstrained testing functions (see Section 5.2). In the second set of experiments, the aim was to evaluate our algorithm using constrained functions (see Section 5.3). In all experiments, to see how the proposed algorithm performs in comparison with other algorithms, the results of MOGOA algorithm were compared with MOPSO [49], NSGA-II [49], and MOALO [55] algorithms.

5.1 Experimental setup

In order to get an unbiased comparison of CPU times, all the experiments are performed using the same PC with the detailed settings as shown in Table 1.

In this section, 12 test problems were selected to evaluate the performance of the proposed MOGOA algorithm. The benchmark functions were divided into two groups, namely, *Constrained* and *Unconstrained*. Each group has six testing functions. In both experiments, the testing functions with diverse characteristics and especially different Pareto optimal Front (PF) were chosen to test the performance of MOGOA from different perspectives. The details of the unconstrained functions are listed in Table 2, while the details of the constrained testing functions are summarized in Section 5.3. For each algorithm, the optimization task was run 10 times for all benchmark problems, and the obtained results are illustrated in the form of *average* $\pm$ *standard deviation*. Moreover, the maximum number of iterations was 100 and the search of agents was 100.

Different assessment methods have been used to evaluate the performance of the proposed algorithm such as: (1) metric of spacing [56], (2) metric of spread [57], and (3) generational distance (GD) [50]. Moreover, for comparison between multiple algorithms and multiple test functions, the average ranks were used. For each given testing function, the algorithms are sorted from best to worst, and the best algorithm receives rank 1, the second best algorithm receives rank 2, and so on. The average ranks are assigned in case of a tie, e.g. if two algorithms tie for the top rank, they both receive rank 1.5. Average ranks of all testing functions are then calculated. Moreover, we used the non-parametric Wilcoxon signed rank test for all of the testing functions to compare different algorithms [58].

5.2 Unconstrained test functions

The goal of this experiment is to evaluate the proposed MOGOA algorithm and compare it with three well-known algorithms MOPSO [49], NSGA-II [49], and MOALO [55]. In this experiment, six unconstrained testing functions were used (ZDT1, ZDT2, ZDT3, ZDT4, ZDT6, and Linear ZDT1 or simply LZDT1); these testing functions are well-known ZDT test suite and it was used in [59]. The first five test functions in this study are identical to those in the original ZDT suite, another last test function is slightly different in the same manner similar to [49]. The details of these functions are summarized in Table 2. Figure 6 shows the obtained Pareto optimal solutions of all algorithms using unconstrained functions. The results of this experiment are summarized in Tables 3, 4, and 5.

Tables 3, 4, and 5 allow us to draw the following conclusions:

Table 1 The detailed settings

Name	Detailed settings
Hardware	
CPU	Core (TM) i5-2400
Frequency	3.10 GHz
RAM	4G
Hard Drive	160 GB
Software	
Operating system	Window 7
Language	MATLAB R2012a (7.14)

Table 2 Unconstrained test functions

Test functions	Functions' details
ZDT	Minimize: $F(f_1(x), f_2(x))$, where $f_1(x) = x_1 0,$ $f_2(x) = g(x).\left(1-\sqrt{\frac{f_1(x)}{g(x)}}\right),$ $g(x) = 1+\frac{9}{N-1}\sum_{i=2}^{N} x_i,$ $0 \le x_i \le 1, 1 \le i \le 30$
ZDT2	Minimize: $F(f_1(x), f_2(x))$, where $f_1(x) = x_1,$ $f_2(x) = g(x).\left(1-(\frac{f_1(x)}{g(x)})^2\right),$ $g(x) = 1+\frac{9}{N-1}\sum_{i=2}^{N} x_i,$ $0 \le x_i \le 1, 1 \le i \le 30$
ZDT3	Minimize: $F(f_1(x), f_2(x))$, where $f_1(x) = x_1,$ $f_2(x) = g(x).$ $\left(1-\sqrt{\frac{f_1(x)}{g(x)}}-(\frac{f_1(x)}{g(x)})\sin(10\pi f_1(x))\right),$ $g(x) = 1+\frac{9}{29}\sum_{i=2}^{N} x_i,$ $0 \le x_i \le 1,\ 1 \le i \le 30$
ZDT4	Minimize: $F(f_1(x), f_2(x))$, where $f_1(x) = x_1,$ $f_2(x) = g(x).\left(1-\sqrt{\frac{f_1(x)}{g(x)}}\right),$ $g(x) = 1+10(n-1)+\sum_{i=2}^{N}(x_i^2-10\cos(4\pi x_i)),$ $0 \le x_1 \le 1,\ -5 \le x_i \le 5,\ 1 \le i \le 10$
ZDT6	Minimize: $F(f_1(x), f_2(x))$, where $f_1(x) = 1-\exp(-4x_1).sin^6(6\pi x_1),$ $f_2(x) = g(x).\left(1-(\frac{f_1(x)}{g(x)})^2\right),$ $g(x) = 1+[\frac{\sum_{i=2}^{n} x_i}{n-1}]^{0.25},$ $0 \le x_i \le 1,\ 1 \le i \le 10$
LZDT1	Minimize: $F(f_1(x), f_2(x))$, where $f_1(x) = x_1,$ $f_2(x) = g(x).\left(1-\frac{f_1(x)}{g(x)}\right),$ $g(x) = 1-\frac{9}{N-1}\sum_{i=2}^{N} x_i,$ $0 \le x_i \le 1,\ 1 \le i \le 30$

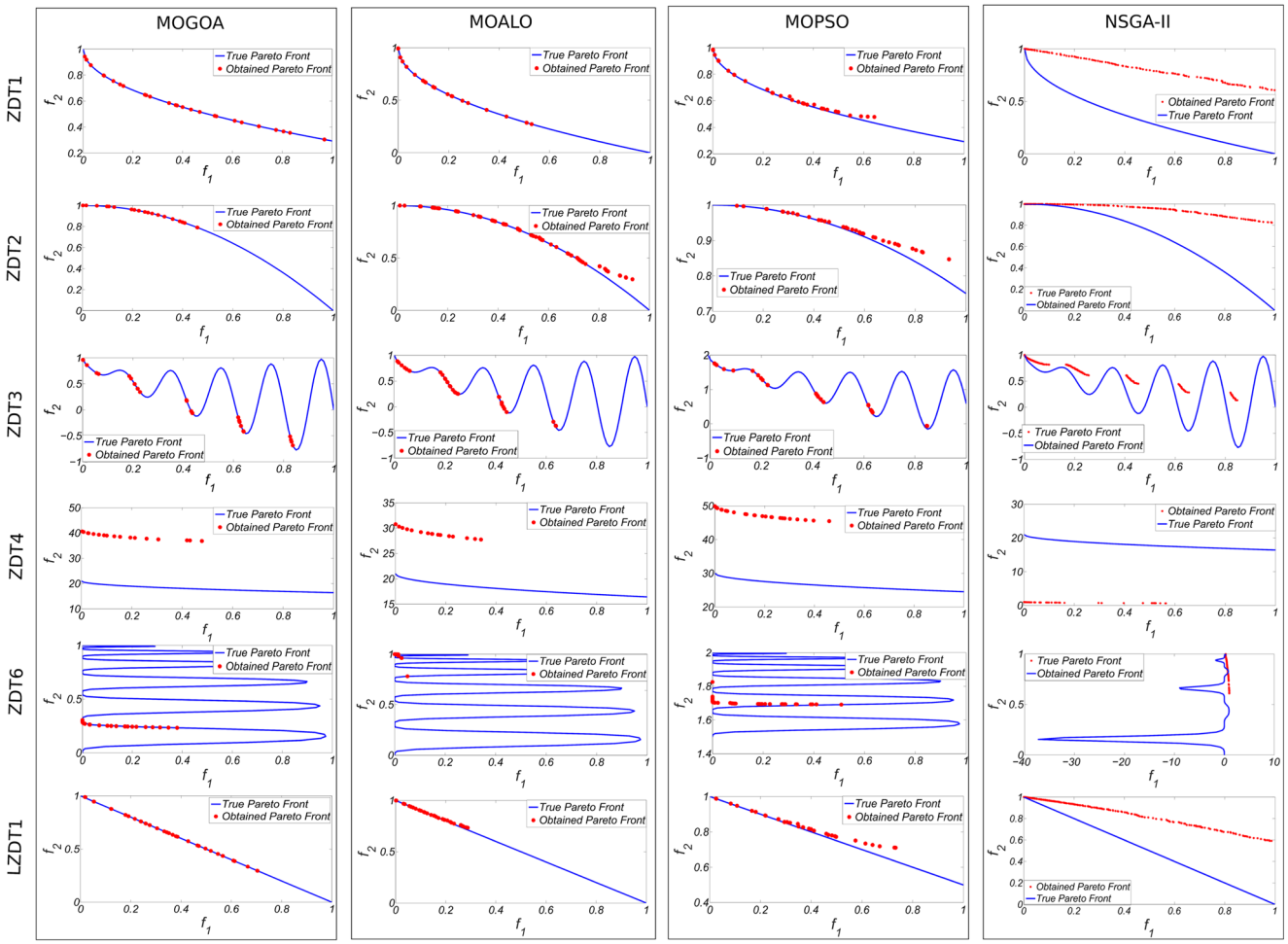

Fig. 6 Obtained Pareto optimal solutions by the MOGOA, MOALO, MOPSO, and NSGA-II algorithms with unconstrained functions

- In terms of the metric of spacing, the MOGOA achieved results better than all the other algorithms in most cases. As shown in Table 3, the MOGOA achieved the best results with all testing functions and it achieved the second best results with ZDT3. In addition, the MOALO and MOPSO obtained the second and third best solutions, respectively. Moreover, from the table, the results with * sign means that the p-value for this algorithm is larger than the predicted statistical significance level of 0.005. As shown, the p-value for ZDT3 with MOPSO algorithm and ZDT4 with MOALO algorithm were greater than the predicted statistical significance level of 0.005, but the other p-values are smaller than the significance level of 0.005. Further, Fig. 7 shows the average ranks for all algorithms and as shown, the MOGOA achieved the lowest, i.e., best, rank.
- In terms of metric of spread results, as shown in Table 4, the MOGOA achieved the best results with

Table 3 Statistical results of metric of spacing with unconstrained testing functions

Testing Function	MOGOA	MOALO	MOPSO	NSGA-II
ZDT1	0.14±0.03	0.17±0.04	0.26±0.15	0.45±0.21
ZDT2	0.17±0.03	0.23±0.09	0.31±0.11	0.56±0.20
ZDT3	0.21±0.11	0.24±0.14	0.20±0.13*	0.34±0.15
ZDT4	0.51±0.32	0.53±0.38*	0.62±0.34	0.60±0.32
ZDT6	0.25±0.19	0.34±0.13	0.46±0.23	0.57±0.23
LZDT1	0.09±0.04	0.12±0.06	0.19±0.11	0.31±0.19

Table 4 Statistical results of metric of spread with unconstrained testing functions

Testing Function	MOGOA	MOALO	MOPSO	NSGA-II
ZDT1	0.15±0.09	0.23±0.12	0.34±0.16	1.1±0.54
ZDT2	0.31±0.15	0.26±0.13*	0.42±0.16	0.85±0.41
ZDT3	0.30±0.21	0.34±0.16	0.36±0.20	0.89±0.35
ZDT4	0.51±0.25	0.67±0.37	0.61±0.23	0.94±0.36
ZDT6	0.49±0.21	0.82±0.34	0.67±0.25	1.2±0.46
LZDT1	0.16±0.03	0.31±0.11	0.42±0.16	0.68±0.31

Table 5 Statistical results of generation distance metric with unconstrained testing functions

Testing Function	MOGOA	MOALO	MOPSO	NSGA-II
ZDT1	0.04±0.01	0.06±0.02	0.09±0.05	0.21±0.13
ZDT2	0.06±0.02	0.09±0.04	0.16±0.08	0.23±0.12
ZDT3	0.09±0.03	0.08±0.03*	0.09±0.04*	0.31±0.09
ZDT4	0.34±0.13	0.32±0.11*	0.42±0.17	0.37±0.16
ZDT6	0.19±0.06	0.21±0.05	0.25±0.12	0.32±0.19
LZDT1	0.03±0.01	0.05±0.02	0.13±0.06	0.23±0.11

all testing functions except with ZDT2 the MOGOA obtained the second best results. Moreover, the MALO obtained the second best results and NSGA-II obtained the worst results. These results are in agreement with Fig. 7. As shown, the MOGOA algorithm obtained the best rank. Additionally, from the table, the p-value for ZDT2 with MOALO algorithm was greater than the predicted statistical significance level of 0.005, but the other p-values are smaller than the significance level of 0.005.

- In terms of GD results, as shown in Table 5, the MOGOA algorithm outperformed the other three algorithms in most cases. As shown, with the ZDT3 and ZDT4 testing functions the MOGOA obtained the second best results and obtained the best results with the other testing functions. Moreover, the MOALO achieved the second best results and the NSGA-II obtained the worst results. Figure 7 shows that the MOGOA algorithm obtained results better than the other algorithms. Further, from Table 5 the p-values for ZDT3 (with MOALO and MOPSO) and ZDT4 (with MOALO) testing functions were greater than the predicted statistical significance level of 0.005, but the other p-values are smaller than the significance level of 0.005.

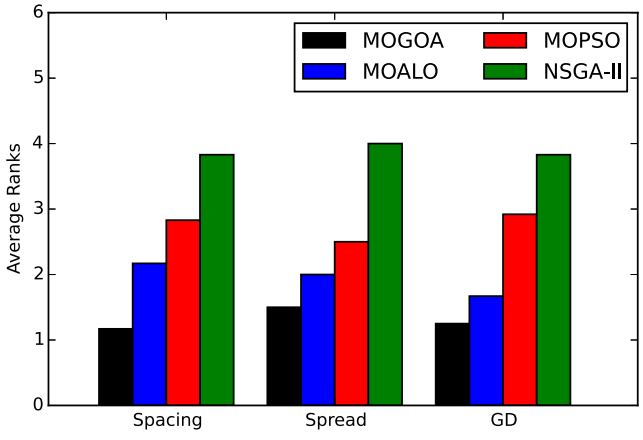

Fig. 7 Average ranking of the comparison between the proposed MOGOA algorithm and the other three algorithms, i.e., MOALO, MOPSO, and NSGA-II, with the unconstrained testing functions

Figure 6 illustrates the best PF obtained (in one run) by MOGOA, MOALO, MOPSO, and NSGA-II algorithms with the unconstrained functions. As shown, the NSGA-II shows the worst convergence which is in agreement with the obtained results. Moreover, the MOGOA, MOALO, and MOPSO algorithms provide a very good convergence toward all true Pareto-optimal fronts.

To conclude, compared with the MOPSO, MOALO, and NSGA-II algorithms, the proposed MOGOA algorithm achieved the best results and a better convergence toward all the true Pareto optimal fronts.

5.3 Constrained test functions

The aim of this experiment is to evaluate the performance of the proposed MOGOA algorithm when six constrained testing functions were used (CONSTR, TNK, SRN, BNH, OSY, and KITA). The results of the MOGOA were compared with MOPSO, NSGA-II, and MOALO algorithms. The details of these functions are summarized in this section. Figure 8 shows the obtained Pareto optimal solutions for all optimization algorithms with all constrained functions. The results of this experiment are summarized in Tables 6, 7, and 8.

The details of the constrained testing functions that were used in our experiments are as follows:

1. **CONSTR**: This function represents a mathematical problem and it has two design variables [34]. The Pareto optimal front for this function is convex and the function is defined as follows:

$$\begin{aligned} Minimize \quad & F(f_1(x), f_2(x)), \text{ where} \\ & f_1(x) = x_1, \\ & f_2(x) = \frac{(1+x_2)}{(x_1)}, \end{aligned} \tag{16}$$

Subject to:

$$\begin{aligned} g_1(x) &= 6 - (x_2 + 9x_1) \leq 0, \\ g_2(x) &= 1 + x_2 - 9x_1 \leq 0, \\ 0.1 \leq x_1 \leq 1, &\, 0 \leq x_2 \leq 5 \end{aligned} \tag{17}$$

2. **TNK**: This function has two design variables and it is defined as in (18). This function has some convex regions and it has discontinuous Pareto optimal front which lies on the boundary of the first constraint [60].

$$\begin{aligned} Minimize \quad & : F(f_1(x), f_2(x)), \text{ where} \\ & f_1(x) = x_1, \\ & f_2(x) = x_2, \end{aligned} \tag{18}$$

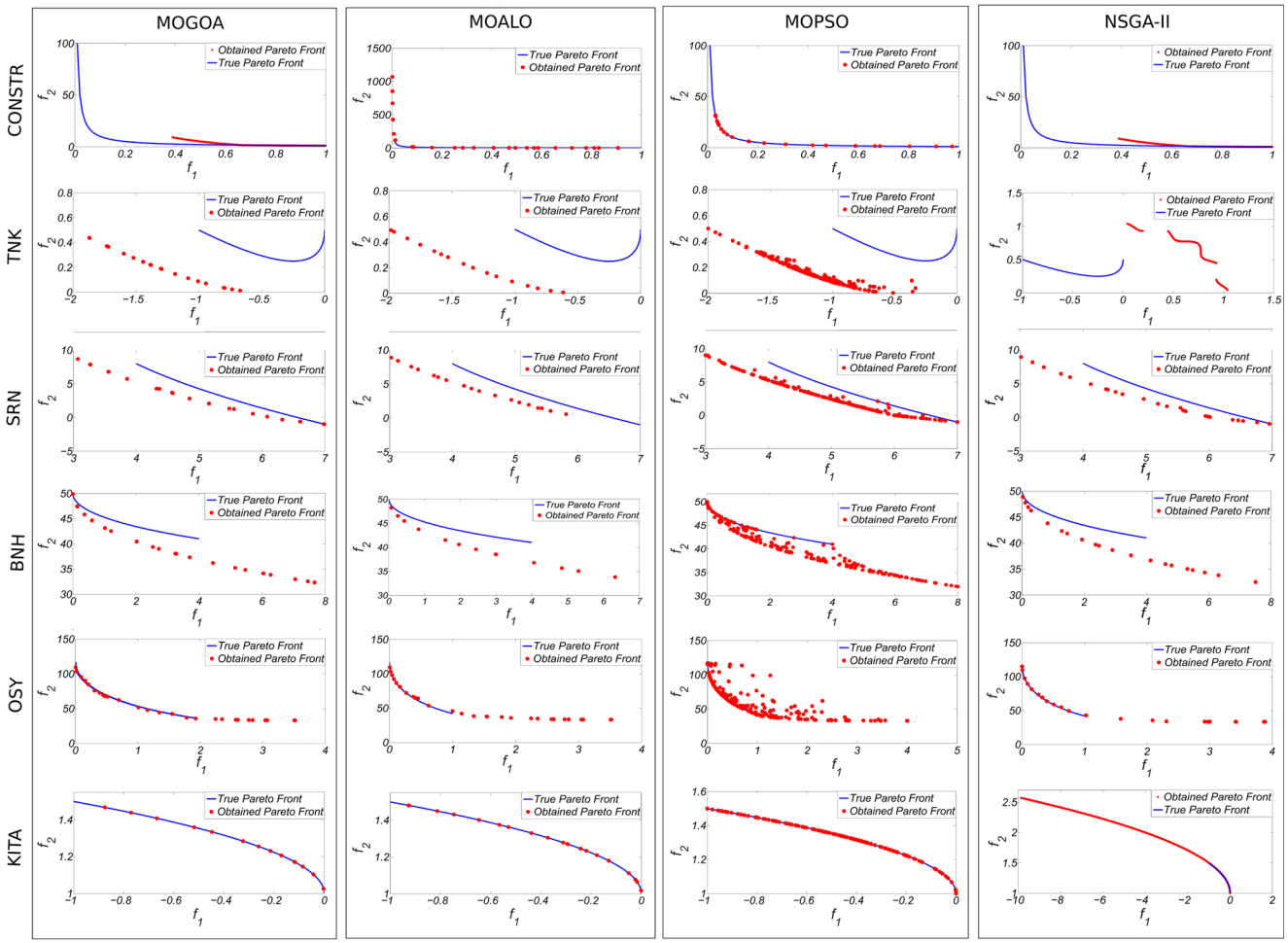

Fig. 8 Obtained Pareto optimal solutions by the MOGOA, MOALO, MOPSO, and NSGA-II algorithms with constrained functions

Subject to:

$$g_1(x) = -x_1^2 - x_2^2 + 1 + 0.1Cos(16arctan(\frac{x_1}{x_2})) \leq 0,$$
$$g_2(x) = 0.5 - (x_1 - 0.5)^2 - (x_2 - 0.5)^2 \leq 0,$$
$$0.1 \leq x_1 \leq \pi, 0 \leq x_2 \leq \pi \quad (19)$$

3. **SRN**: This function is introduced by Srinivas and Deb in [57]. It has two design variables with a continuous Pareto optimal front. It is given by:

$$Minimize \ : F(f_1(x), f_2(x)), \text{where}$$
$$f_1(x) = 2 + (x_1 - 2)^2 + (x_2 - 1)^2,$$
$$f_2(x) = 9x_1 - (x_2 - 1)2, \quad (20)$$

Subject to:

$$g_1(x) = x_1^2 + x_2^2 - 255 \leq 0,$$
$$g_2(x) = x_1 - 3x_2 + 10 \leq 0,$$
$$-20 \leq x_1 \leq 20, -20 \leq x_2 \leq 20 \quad (21)$$

Table 6 Statistical results of metric of spacing with constrained testing functions

Testing Function	MOGOA	MOALO	MOPSO	NSGA-II
CONSTR	$2.29e^{-2} \pm 2.0e^{-3}$	2.14e-2±$2.7e^{-3}$*	3.25e-2±$2.4e^{-3}$	$4.37e^{-2} \pm 4.1e^{-3}$
TNK	$1.6e^{-2} \pm 1.2e^{-3}$	$0.2e^{-2} \pm 0.1e^{-3}$*	2.17e-2±$1.9e^{-3}$	3.42e-2±$2.6e^{-3}$
SRN	0.61±0.13	0.70±0.10	1.28±0.2	1.59±0.13
BNH	0.31±$3.6e^{-2}$	0.34±$2.4e^{-2}$	0.69±$3.8e^{-2}$	0.78±$7.2e^{-2}$
OSY	0.39±$6.3e^{-2}$	0.49±$7.6e^{-2}$	0.52±$9.5e^{-2}$	1.14±$2.8e^{-2}$
KITA	0.26±0.39	0.29±0.42*	0.32±0.48*	0.44±0.15

Table 7 Statistical results of metric of spread with constrained testing functions

Testing Function	MOGOA	MOALO	MOPSO	NSGA-II
CONSTR	$0.29\pm2.0e^{-2}$	$3.46\pm1.0e^{-2}$	$0.94\pm3.7e^{-1}$	$0.55\pm2.7e^{-2}$
TNK	$0.63\pm3.3e^{-2}$	$0.64\pm1.2e^{-2}$	$0.79\pm5.1e^{-2}$	$0.82\pm2.9e^{-4}$
SRN	$0.53\pm5.1e^{-2}$	$0.39\pm2.5e^{-2*}$	$0.67\pm7.2e^{-2}$	$0.39\pm2.5e^{-2}$
BNH	$0.36\pm1.1e^{-2}$	$0.37\pm2.6e^{-2*}$	$0.48\pm2.7e^{-2}$	$0.53\pm2.4e^{-2}$
OSY	$0.34\pm2.7e^{-2}$	$0.39\pm2.6e^{-2}$	$0.49\pm1.9e^{-2}$	$0.62\pm2.8e^{-2}$
KITA	0.52±0.15	0.60±0.19	0.99±0.12	0.79±0.20

4. **BNH**: This function was first introduced by Binh and Korn in [61], and it is defined as follow:

$$\begin{aligned} Minimize: \quad & F(f_1(x), f_2(x)), \text{ where} \\ & f_1(x) = 4x_1^2 + 4x_2^2, \\ & f_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2, \end{aligned} \tag{22}$$

Subject to:

$$\begin{aligned} & g_1(x) = (x_1 - 5)^2 + x_2^2 - 25 \leq 0, \\ & g_2(x) = 7.7 - (x_1 - 8)^2 - (x_2 + 3)^2 \leq 0, \\ & 0 \leq x_1 \leq 5, 0 \leq x_2 \leq 3 \end{aligned} \tag{23}$$

5. **OSY**: The OSY function has five separated regions and it was proposed by Osyczka and Kundu [62]. Moreover, it has six constraints and six design variables. The definition of this variable is as follows:

$$\begin{aligned} Minimize: \quad & F(f_1(x), f_2(x)), \text{ where} \\ & f_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2, \\ & f_2(x) = [25(x_1 - 2)^2 + (x_2 - 1)^2 + (x_3 - 1) \\ & +(x_4 - 4)^2 + (x_5 - 1)^2], \end{aligned} \tag{24}$$

Subject to:

$$\begin{aligned} & g_1(x) = 2 - x_1 - x_2 \leq 0, \\ & g_2(x) = -6 + x_1 + x_2 \leq 0, \\ & g_3(x) = -2 - x_1 + x_2 \leq 0, \\ & g_4(x) = -2 + x_1 - 3x_2 \leq 0, \\ & g_5(x) = -4 + x_4 + (x_3 - 3)^2 \leq 0, \\ & g_6(x) = 4 - x_6 - (x_5 - 3)^2 \leq 0, \\ & 0 \leq x_1 \leq 10, 0 \leq x_2 \leq 10, 1 \leq x_3 \leq 5, \\ & 0 \leq x_4 \leq 6, 1 \leq x_5 \leq 5, 0 \leq x_6 \leq 10 \end{aligned} \tag{25}$$

6. **KITA**: This function was introduced by Kita et al. [63], and it has been widely used. The mathematical formula for this function is as follows:

$$\begin{aligned} Maximize \quad : \quad & F(f_1(x), f_2(x)), \text{ where} \\ & f_1(x) = -x_1^2 + x_2, \\ & f_2(x) = \frac{1}{2}x_1 + x_2 + 1, \end{aligned} \tag{26}$$

Subject to:

$$\begin{aligned} & g_1(x) = \frac{1}{6}x_1 + x_2 - \frac{13}{2} \leq 0 \\ & g_2(x) = \frac{1}{2}x_1 + x_2 - \frac{15}{2} \leq 0 \\ & g_3(x) = 5x_1 + x_2 - 30 \leq 0 \\ & 0 \leq x_1, x_2 \leq 7 \end{aligned} \tag{27}$$

From Tables 6, 7, and 8, the following notes can be remarked:

- In terms of the metric of spacing, the MOGOA obtained results better than all the other algorithms in most cases. As shown in Table 6, the MOGOA achieved the best results with all testing functions and it achieved the second best results with the CONSTR and TNK functions. Additionally, the MOALO and MOPSO obtained the second and third best solutions, respectively. Moreover, from the table, the results with * sign means that the p-value for this algorithm is larger than the predicted statistical significance level of 0.005 (as in the first experiment). As shown, the p-value for CONSTR, TNK, and KITA with MOALO algorithm were greater

Table 8 Statistical results of generation distance metric with constrained testing functions

Testing Function	MOGOA	MOALO	MOPSO	NSGA-II
CONSTR	$1.37e^{-3} \pm 3.6e^{-5}$	$1.7e^{-4} \pm 4.6e^{-5}$	$4.54e^{-3} \pm 6.89e^{-4}$	$5.14e^{-3} \pm 2.5e^{-4}$
TNK	$4.86e^{-4} \pm 3.6e^{-5}$	$7.97e^{-4} \pm 5.4e^{-5}$	$5.09e^{-3} \pm 4.6e^{-4}$	$4.05e^{-3} \pm 4.4e^{-4}$
SRN	$4.36e^{-5} \pm 2.1e^{-4}$	$6.89e^{-5} \pm 3.5e^{-6}$	$2.76e^{-3} \pm 2.1e^{-4}$	$3.71e^{-3} \pm 5.1e^{-4}$
BNH	$2.05e^{-4} \pm 5.7e^{-5}$	$3.15e^{-3} \pm 3.5e^{-5}$	$4.62e^{-3} \pm 2.9e^{-5}$	$4.91e^{-3} \pm 2.8e^{-5}$
OSY	$3.10e^{-2} \pm 2.6e^{-2}$	$3.27e^{-2} \pm 2.5e^{-2}$	$9.68e^{-2} \pm 7.2e^{-2}$	$9.89e^{-1} \pm 9.78e^{-1}$
KITA	$3.91e^{-2} \pm 4.7e^{-2}$	$4.20e^{-2} \pm 4.9e^{-2}$	$4.67e^{-2} \pm 5.4e^{-2}$	$4.00e^{-2} \pm 4.4e^{-2}$

than the predicted statistical significance level of 0.005, but the other p-values are smaller than the significance level of 0.005. Further, Fig. 9 shows the average ranks for all algorithms and as shown the MOGOA achieved the lowest, i.e., best, rank, and the NSGA-II algorithm was the worst one.

- In terms of metric of spread results, as shown in Table 7, the MOGOA obtained the best results with all testing functions except with SNR function the MOGOA obtained the second best results. Moreover, the MOALO obtained the second best results in most cases and the NSGA-II algorithm obtained the worst results. These results are in agreement with Fig. 9. As shown, the MOGOA algorithm obtained the best rank and the NSGA-II attained the highest, i.e., worst, rank. Additionally, from the table, the p-value for SRN and BNH testing functions with MOALO algorithm was greater than the predicted statistical significance level of 0.005, but the other p-values are smaller than the significance level of 0.005.
- In terms of GD results, as shown in Table 8, the MOGOA algorithm outperformed the other three algorithms in all cases, and the MOALO achieved the second best results, while the MOPSO and NSGA-II algorithms obtained the worst results. Figure 9 displays that the MOGOA algorithm obtained results better than the other algorithms. Further, from Table 8 the p-values for SNR and BNH testing functions with MOALO algorithm were greater than the predicted statistical significance level of 0.005, but the other p-values are smaller than the significance level of 0.005.

Figure 8 displays the PF obtained (in one run) by MOGOA, MOALO, MOPSO, and NSGA-II algorithms with the constrained functions. As shown, the constrained test functions have very different Pareto fronts compared with the unconstrained test functions, such as CONSTR, BNH, and OSY. CONSTR function has a concave front attached to a linear front. As shown, the MOGOA and MOALO managed to approximate the CONSTR function successfully. The OSY function is slightly similar to CONSTR function but with multiple linear regions with different slopes. Moreover, the TNK function has a wave-shaped front. As shown in Fig. 8, the MOGOA and MOALO algorithms provide a very good convergence toward most of the true Pareto-optimal fronts.

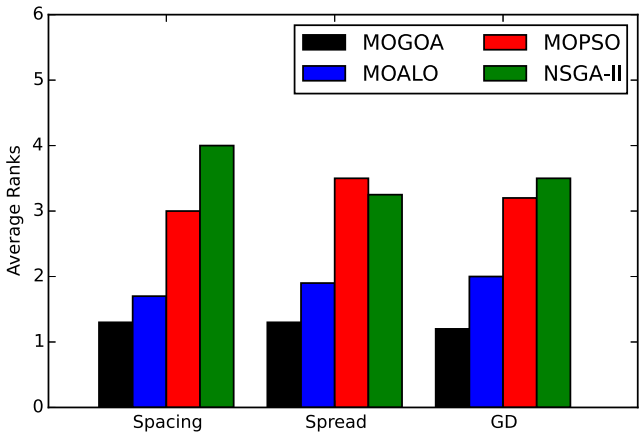

Fig. 9 Average ranking of the comparison between the proposed MOGOA algorithm and the other three algorithms, i.e., MOALO, MOPSO, and NSGA-II, with the constrained testing functions

Generally, the proposed MOGOA algorithm obtained the best results and a competitive convergence toward all the true Pareto optimal fronts compared with the MOPSO, MOALO, and NSGA-II algorithms.

6 Conclusions

This paper proposed a multi-objective version of the recently proposed Grasshopper optimization algorithm (GOA) called MOGOA. The proposed algorithm was designed by integrating the GOA with an external archive and grasshopper selection mechanism based on Pareto optimal dominance. The goal of the external archive is to keep non-dominated solutions. The proposed MOGOA utilized the same features of the GOA. The MOGOA was verified by 12 testing functions including six unconstrained functions and six constrained functions. In our experiments, three assessment methods were used: generational distance metric, metric of spacing, and metric of spread. The findings of our experiments proved that the proposed MOGOA was able to find the optimal Pareto front (PF) and provide a superior quality of solutions in comparison with a variety of other algorithms such as Multi-Objective Particle Swarm Optimization (MOPSO), Multi-Objective Ant Lion Optimizer (MOALO), and Non-dominated Sorting Genetic Algorithm version 2 (NSGA-II). In general, according to the reported results, the MOGOA offers competitive solutions compared with the other multi-objective algorithms and it offers a wider range of non-dominated solutions.

For future studies, we are planning to employ the MOGOA algorithm in machine learning-related applications. In this area, some applications have many problems with different objectives such as feature selection, parameter optimization, i.e., parameter tuning, and data preprocessing. Our next goal is to employ the MOGOA in such problems. Moreover, different modifications will be added to the MOGOA such as using Chaotic maps, this is called Chaotic optimization, for generating values for c parameter. This modification can help the MOGOA to converge to the optimal solution faster than standard stochastic search as reported in [64].

References

1. Motevasel M, Seifi AR, Niknam T (2013) Multi-objective energy management of chp (combined heat and power)-based micro-grid. Energy 51:123–136
2. Elhoseny M, Tharwat A, Hassanien AE (2017) Bezier curve based path planning in a dynamic field using modified genetic algorithm. J Comput Sci, In Press
3. Tharwat A, Gabel T, Hassanien AE (2017) Parameter optimization of support vector machine using dragonfly algorithm. In: International conference on advanced intelligent systems and informatics. Springer, Berlin, pp 309–319
4. Elhoseny M, Tharwat A, Farouk A, Hassanien AE (2017) K-coverage model based on genetic algorithm to extend wsn lifetime. IEEE Sensors Lett 1(4):1–4
5. Handl J, Kell DB, Knowles J (2007) Multiobjective optimization in bioinformatics and computational biology. IEEE/ACM Trans Comput Biol Bioinform 4(2):279–292
6. Kipouros T, Jaeggi DM, Dawes WN, Parks GT, Savill AM, Clarkson PJ (2008) Biobjective design optimization for axial compressors using tabu search. AIAA J 46(3):701
7. Tharwat A, Gabel T, Hassanien AE (2017) Classification of toxicity effects of biotransformed hepatic drugs using optimized support vector machine. In: International conference on advanced intelligent systems and informatics. Springer, Berlin, pp 161–170
8. Hassanien AE, Tharwat A, Own HS (2017) Computational model for vitamin d deficiency using hair mineral analysis. Comput Biol Chem 70:198–210
9. Rizk-Allah RM, Hassanien AE (2017) A hybrid optimization algorithm for single and multi-objective optimization problems. In: Handbook of research on machine learning innovations and trends. IGI Global, Hershey, pp 491–521
10. Marler RT, Arora JS (2004) Survey of multi-objective optimization methods for engineering. Struct Multidiscip Optim 26(6):369–395
11. Deb K (2012) Advances in evolutionary multi-objective optimization. In: Search based software engineering, pp 1–26
12. Coello CAC (2009) Evolutionary multi-objective optimization: some current research trends and topics that remain to be explored. Front Comput Sci Chin 3(1):18–30
13. Coello CAC, Lamont GB, Van Veldhuizen DA et al (2007) Evolutionary algorithms for solving multi-objective problems, 2nd edn. Springer, Berlin
14. Padhye N, Mittal P, Deb K (2015) Feasibility preserving constraint-handling strategies for real parameter evolutionary optimization. Comput Optim Appl 62(3):851–890
15. Coello CC (2006) Evolutionary multi-objective optimization: a historical view of the field. IEEE Comput Intell Mag 1(1):28–36
16. Padhye N, Bhardawaj P, Deb K (2013) Improving differential evolution through a unified approach. J Glob Optim 55(4):771
17. Deb K, Padhye N (2014) Enhancing performance of particle swarm optimization through an algorithmic link with genetic algorithms. Comput Optim Appl 57(3):761–794
18. Srinivas N, Deb K (1994) Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evol Comput 2(3):221–248
19. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans Evol Comput 6(2):182–197
20. Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans Evol Comput 3(4):257–271
21. Coello CAC, Pulido GT, Lechuga MS (2004) Handling multiple objectives with particle swarm optimization. IEEE Trans Evol Comput 8(3):256–279
22. Padhye N, Branke J, Mostaghim S (2009) Empirical comparison of mopso methods-guide selection and diversity preservation. In: IEEE congress on evolutionary computation (CEC'09). IEEE, New York, pp 2516–2523
23. Padhye N (2009) Comparison of archiving methods in multiobjectiveparticle swarm optimization (mopso): empirical study. In: Proceedings of the 11th annual conference on genetic and evolutionary computation. ACM, New York, pp 1755–1756
24. Zhang Q, Li H (2007) Moea/d: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11(6):712–731
25. Abbass HA, Sarker R, Newton C (2001) Pde: a Pareto-frontier differential evolution approach for multi-objective optimization problems. In: Proceedings of the 2001 congress on evolutionary computation, vol 2. IEEE, New York, pp 971–978
26. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
27. Saremi S, Mirjalili S, Lewis A (2017) Grasshopper optimisation algorithm: theory and application. Adv Eng Softw 105:30–47
28. Haupt RL, Haupt SE (2004) Practical genetic algorithms. Wiley, New York
29. 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 264
30. Ščap D, Hoić M, Jokić A (2013) Determination of the Pareto frontier for multiobjective optimization problem. Transactions of FAMENA 37(2):15–28
31. Kim IY, de Weck OL (2005) Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. Struct Multidiscip Optim 29(2):149–158
32. Das I, Dennis JE (1998) Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM J Optim 8(3):631–657
33. Parsopoulos KE, Vrahatis MN (2002) Particle swarm optimization method in multiobjective problems. In: Proceedings of the 2002 ACM symposium on applied computing. ACM, New York, pp 603–607
34. Deb K (2011) Multi-objective optimisation using evolutionary algorithms: an introduction. In: Multi-objective evolutionary optimisation for product design and manufacturing. Springer, Berlin, pp 3–34
35. Goldberg D (1989) Genetic algorithms in optimization, search and machine learning. Addison-Wesley, Reading
36. Tharwat A, Gaber T, Hassanien AE, Elnaghi BE (2017) Particle swarm optimization: a tutorial. In: Handbook of research on machine learning innovations and trends. IGI Global, Hershey, pp 614–635
37. Nebro AJ, Durillo JJ, Coello CAC (2013) Empirical comparison of mopso methods-guide selection and diversity preservation. In: IEEE congress on evolutionary computation (CEC). IEEE, New York, pp 3153–3160
38. Knowles J, Thiele L, Zitzler E (2006) A tutorial on the performance assessment of stochastic multiobjective optimizers. Tik Report 214:327–332
39. Pradhan PM, Panda G (2012) Solving multiobjective problems using cat swarm optimization. Expert Syst Appl 39(3):2956–2964
40. Shi X, Kong D (2015) A multi-objective ant colony optimization algorithm based on elitist selection strategy. Metallurgical & Mining Industry 7(6):333–338
41. Hancer E, Xue B, Zhang M, Karaboga D, Akay B (2015) A multi-objective artificial bee colony approach to feature selection using

fuzzy mutual information. In: IEEE congress on evolutionary computation (CEC). IEEE, New York, pp 2420–2427
42. Hemmatian H, Fereidoon A, Assareh E (2014) Optimization of hybrid laminated composites using the multi-objective gravitational search algorithm (mogsa). Eng Optim 46(9):1169–1182
43. Velazquez JMO, Coello CAC, Arias-Montano A (2014) Multi-objective compact differential evolution. In: IEEE symposium on differential evolution (SDE). IEEE, New York, pp 1–8
44. Yamany W, El-Bendary N, Hassanien AE, Emary E (2016) Multi-objective cuckoo search optimization for dimensionality reduction. Procedia Computer Science 96:207–215
45. Emary E, Yamany W, Hassanien AE, Snasel V (2015) Multi-objective gray-wolf optimization for attribute reduction. Procedia Computer Science 65:623–632
46. Lin W, Yu D, Wang S, Zhang C, Zhang S, Tian H, Luo M, Liu S (2015) Multi-objective teaching–learning-based optimization algorithm for reducing carbon emissions and operation time in turning operations. Eng Optim 47(7):994–1007
47. Coello CA (2000) An updated survey of ga-based multiobjective optimization techniques. ACM Comput Surv (CSUR) 32(2):109–143
48. Pareto V (1964) Cours d'économie politique, vol 1. Librairie Droz
49. Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Applic 27(4):1053–1073
50. Van Veldhuizen DA, Lamont GB (1998) Multiobjective evolutionary algorithm research: a history and analysis. Tech. rep., Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB Ohio
51. Coello CC, Pulido GT (2005) Multiobjective structural optimization using a microgenetic algorithm. Struct Multidiscip Optim 30(5):388–403
52. Sadollah A, Eskandar H, Kim JH (2015) Water cycle algorithm for solving constrained multi-objective optimization problems. Appl Soft Comput 27:279–298
53. Tharwat A, Hassanien AE, Elnaghi BE (2016) A ba-based algorithm for parameter optimization of support vector machine. Pattern Recogn Lett 93:13–22
54. Tharwat A, Elnaghi BE, Hassanien AE (2016) Meta-heuristic algorithm inspired by grey wolves for solving function optimization problems. In: International conference on advanced intelligent systems and informatics. Springer, Berlin, pp 480–490
55. Mirjalili S, Jangir P, Saremi S (2017) Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems. Appl Intell 46(1):79–95
56. Schott JR (1995) Fault tolerant design using single and multicriteria genetic algorithm optimization. Tech. rep., DTIC Document
57. Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, Chicheter
58. García S, Fernández A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. Inform Sci 180(10):2044–2064
59. Zitzler E, Deb K, Thiele L (2000) Comparison of multiobjective evolutionary algorithms: empirical results. Evol Comput 8(2):173–195
60. Tanaka M, Watanabe H, Furukawa Y, Tanino T (1995) Ga-based decision support system for multicriteria optimization. In: IEEE international conference on systems, man and cybernetics. Intelligent systems for the 21st century, vol 2. IEEE, New York, pp 1556–1561
61. Binh TT, Korn U (1997) Mobes: a multiobjective evolution strategy for constrained optimization problems. In: The third international conference on genetic algorithms (Mendel 97), vol 25, p 27
62. Osyczka A, Kundu S (1995) A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. Struct Multidiscip Optim 10(2):94–99
63. Kita H, Yabumoto Y, Mori N, Nishikawa Y (1996) Multi-objective optimization by means of the thermodynamical genetic algorithm. In: Parallel problem solving from nature—PPSN IV, pp 504–512
64. Tharwat A, Hassanien AE (2017) Chaotic antlion algorithm for parameter optimization of support vector machine. Appl Intell 1–17, In Press

Alaa Tharwat received his BSc in 2002 from Mansoura University, MSc in 2008 Mansoura University, and PhD in 2017 from Suez Canal University. He worked as an assistant lecturer at Gent University, within the framework of the Welcome project–Erasmus Mundus Action 2 – with a title 'Novel approach of multimodal biometrics for animal identification' in 2015. Currently, he is an assistant lecturer in Faculty of Computer Science and Engineering, Frankfurt University of Applied Sciences, Frankfurt am Main, Germany. He has more than 30 scientific research papers published in prestigious international journals in the topics of machine learning and its applications.

Essam H. Houssein received his Ph.D. degree in "Studying Some Routing Problems of Mobile Ad Hoc Networks using Artificial Ant Colony", 2012. Essam Halim Houssein is the vice chair of the Scientific Research Group in Egypt (SRGE) for the Scientific Affairs and Conferences, http://egyptscience.net/. Currently work as Lecturer at CS Dept., Faculty of Computers and Information, Minia University, EGYPT. Essam has more than 30 scientific research papers published in prestigious international journals covering such diverse topics as data mining, medical images, intelligent systems, and networks. His research interests include wireless sensor networks, cloud computing, security, soft computing, Image processing, optimization, metaheuristics.

Mohammed M. Ahmed was born in Minia, Egypt in 1989. He received his B.Sc. degree in 2010 from Faculty of Computers and Information, Information System Department, Minia University, Egypt. And he received M.Sc degree from FCI-Cairo University, Information System Department, Egypt.

Aboul Ella Hassanien received his B.Sc. with honors in 1986 and M.Sc. degree in 1993, both from Ain Shams University, Faculty of Science, Pure Mathematics and Computer Science Department, Cairo, Egypt. On September 1998, he received his doctoral degree from the Department of Computer Science, Graduate School of Science & Engineering, Tokyo Institute of Technology, Japan. He is the Founder and Head of the Egyptian Scientific Research Group (SRGE) and Professor of Information Technology at the Faculty of Computer and Information, Cairo University. He has more than 500 scientific research papers published in prestigious international journals and he has more than 30 books in the topics of data mining and medical images and intelligent systems address social networks and smart environment.

Thomas Gabel studied Computer Science at Technical University of Kaiserslautern, Germany, and at Carnegie Mellon University, USA, and received his Diploma degree (comparable to Master) in 2003. During his PhD studies, he worked at Karlsruhe Institute of Technology and at the University of Osnabrück where he received his PhD in 2009. Subsequently, he joined the Machine Learning Lab at the University of Freiburg as a post-doctoral research fellow. Thomas' experience also includes engagements at IBM Germany and at the German Air Navigation Service Provider (DFS, 2011-14) where he worked as researcher and software engineer in the context of air traffic scheduling and optimization. In September 2014, Thomas joined Frankfurt University of Applied Sciences as professor at the Faculty of Computer Science and Engineering. His main research interests are learning agents, multi-agent systems, reinforcement learning, case-based reasoning, and robotic soccer. As team leader of a robotic soccer simulation team he achieved three world-champion and two vice world-champion titles in RoboCup competitions.