Neural Computing and Applications (2020) 32:3101–3115
https://doi.org/10.1007/s00521-018-3808-3

ORIGINAL ARTICLE

MOGSABAT: a metaheuristic hybrid algorithm for solving multi-objective optimisation problems

Iraq Tariq[1,2] · H. A. AlSattar[3] · A. A. Zaidan[3] · B. B. Zaidan[3] · M. R. Abu Bakar[2] · R. T. Mohammed[4] · O. S. Albahri[3] · M. A. Alsalem[3] · A. S. Albahri[3]

Received: 14 November 2017 / Accepted: 5 October 2018 / Published online: 16 October 2018

Abstract
This study proposes a novel strength of multi-objective gravitational search algorithm and bat algorithm *MOGSABAT* to solve multi-objective optimisation problem. The proposed *MOGSABAT* algorithm is divided into three stages. In the first stage (moving space), a switch in a solution from single function to multiple functions that contain more than one objective to use the gravitational search algorithm *GSA* is determined. We established a new equation to calculate the masses of individuals in the population using the theoretical work found in the strength Pareto evolutionary algorithm. In the second stage (moving in space), how to handle the bat algorithm *BAT* to solve multiple functions is established. We applied the theoretical work of multi-objective particle swarm optimisation into the *BAT* algorithm to solve multiple functions. In the third stage, multi-objective *GSA* and multi-objective *BAT* are integrated to obtain the hybrid *MOGSABAT* algorithm. *MOGSABAT* is tested by adopting a three-part evaluation methodology that (1) describes the benchmarking of the optimisation problem (*bi*-objective and *tri*-objective) to evaluate the performance of the algorithm; (2) compares the performance of the algorithm with that of other intelligent computation techniques and parameter settings; and (3) evaluates the algorithm based on mean, standard deviation and Wilcoxon signed-rank test statistic of the function values. The optimisation results and discussion confirm that the *MOGSABAT* algorithm competes well with advanced metaheuristic algorithms and conventional methods.

Keywords Multi-objective optimisation problem · Gravitational search algorithm · Bat algorithm · Swarm intelligence

1 Introduction

Optimisation problems generally have more than one objective, which is a common nature in many industrial and academic research sectors. Such problems with

✉ A. A. Zaidan
aws.alaa@fskik.upsi.edu.my

Iraq Tariq
iraqtariq70@gmail.com

H. A. AlSattar
hassan.alsattar@gmail.com

B. B. Zaidan
bilalbahaa@fskik.upsi.edu.my

M. R. Abu Bakar
mrizam@upm.edu.my

R. T. Mohammed
rawiatahrirs@gmail.com

O. S. Albahri
osamahsh89@gmail.com

M. A. Alsalem
mohammed.asum@gmail.com

A. S. Albahri
ahmed.bahri1978@gmail.com

[1] Department of Mathematics, Faculty of Science, University of Baghdad, Baghdad, Iraq

[2] Department of Mathematics, FS, Universiti Putra Malaysia, Seri Kembangan, Malaysia

[3] Department of Computing, FSKIK, Universiti Pendidikan Sultan Idris, Tanjong Malim, Malaysia

[4] Department of Computer Science, FSKTM, Universiti Putra Malaysia, Seri Kembangan, Malaysia

conflicted and incommensurable objectives are called multi-objective optimisation problems *MOPs*. In contrast to the single-objective optimisation problem *SOO*, *MOPs* have no single optimum solution, but they have a set of trade-off solutions known as Pareto optimal solutions [33]. As the search space grows dramatically with problem size, the high-dimensional search space in *MOPs* caused the traditional optimisation algorithms using exact techniques (e.g. exhaustive search) to be no longer suitable.

The recent decades showed an interesting and significant growth of algorithms as researchers fully utilised natural phenomenon behaviours as an inspiration technique [9]. These algorithms show good performance in solving complex computational problems with high-dimensional objectives (i.e. *MOPs*). Although various heuristic algorithms have been proposed and show an efficient and effective performance, such as ant colony search algorithm [7], artificial bee colony algorithm [15], genetic algorithm [27], bat algorithm *BAT* [19], particle swarm optimisation *PSO* [11], simulated annealing *SA* [12], gravitational search algorithm *GSA* [21], their achievements cannot be considered as the best in solving all of the *MOPs*. Hence, the need for new optimisation algorithms to address *MOPs* remains a challenge [21].

Basically, exploration and exploitation (also referred to as diversification and intensification, respectively) are two main aspects of population-based heuristic algorithms, where the balance between them in any metaheuristic algorithm is the performance measurement of its success in solving each given MOP. Exploration is the capability to search the space that allows the metaheuristic algorithm to scan the expanding parts of the search space without falling into local optima. Meanwhile, exploitation is the capability to search locally in the search space to provide accurate search and convergence [22].

Although population-based search algorithms have achieved good performance results [29], none of the metaheuristic algorithms have superior performance in solving all problems. Practically, the performance of the algorithm in solving *MOPs* might be controversial from one problem to another. Thus, developing a hybrid metaheuristic algorithm by combining different metaheuristic concepts can improve the quality of performance and meet the promising balance between diversity and convergence [14].

Many previous researchers have proposed algorithms in an attempt to obtain multi-Pareto optimal solutions with an efficient exploitation and global diverse exploration such as *BAT* algorithm that mimic the bat behaviour of echolocation capability. Although *BAT* algorithm shows good results in terms of quality performance for exploration and exploitation, it shows bad convergence and less accuracy performance in some multi-dimensional functions because of trapping in local minimum in some cases [22]. The *GSA*, which is derived from Newton's law on gravity and mass interactions, performed effectively in *SOO* and has shown promising results in *MOPs*, as presented by [8] who proposed a multi-objective *GSA* (*MOGSA*) that shows a good reduction in diversity of population at a search space.

Similar to the particle swarm optimisation [2], the *GSA* has two key points that need to be addressed in tackling objective optimisation problems. The first is related to leaders selection of the population. The second is related to methods on sustaining the obtained good alternative results [26]. Some recent works have attempted to handle these problems by using an external archive for selecting leaders [8], whereas others depended on non-dominated sorting to select leaders from the external archive and population [17]. To address these issues and improve the balance between the diversity and convergence in *MOPs*, we proposed a hybrid metaheuristic algorithm called multi-objective GSA and bat algorithm *MOGSABAT*.

The main contributions of this paper are as follows. Firstly, we established a new equation to calculate the masses of individuals in the population using the theoretical work found in the strength Pareto evolutionary algorithm two (*SPEAII*) [34] to switch in a solution from single function to multiple functions that contain more than one objective to use the *GSA*. Secondly, we applied the theoretical work of multi-objective particle swarm optimisation *MOPSO* in multiple functions into the BAT algorithm to solve multiple functions to handle the bat algorithm. Thirdly, we examined the integration of multi-objective GSA and multi-objective BAT to obtain the hybrid proposed *MOGSABAT* algorithm.

The proposed *MOGSABAT* algorithm is compared with five state-of-the-art techniques to optimise three common MOP suites, which are *ZDT*, *UF* and *BT*. Also, two well-known performance metrics generational distance (*GD*) and reversed *GD* (*RGD*) are statistically applied in this study to test the performance quality of *MOGSABAT* in comparison with other algorithms.

We organised the remainder of this paper as follows. In Sect. 2, we provide overviews on the theoretical background of the proposed *MOGSABAT* algorithm. In Sect. 3, we demonstrated the evaluation methodology. In Sect. 4, we provided the statistical measurement based on mean, standard deviation (*STD*) and Wilcoxon signed-rank test. Finally, in Sect. 5, we draw our conclusions.

2 Methodology

The present study aims to solve the following types of problem (without loss of generality, the present study will only be assuming minimisation problems):

$$\text{Min}\, Z = (f_1(x), f_2(x), \ldots, f_k(x)) \tag{1}$$

where $g_i(x) \geq 0,\ i = 1, 2, \ldots, p;\ h_j(x) = 0,\ j = 1, 2, \ldots, q;$ $x_i^l \leq x_i \leq x_i^u,\ i = 1, 2, \ldots, n;$ $k =$ no. of objective functions, $p =$ no. of inequality constraint, $q =$ no. of the equality constraint. and f, g, h are linear, quadratic or higher functions. Multiple-objective optimisation programming of the form Eq. (1) arises when rates including the ratios profit/revenue, profit/time, raw materials wastage/used raw material quantities are to be maximised. These problems are frequently linear or at least concave/convex fractional programming algorithmic schemes that provide successful solutions to numerous actual optimisation problems, which are naturally *NP*-Hard [23].

To describe the target concept of optimality, the researcher introduces the following definitions [23].

1. **Definition 1** (*Pareto dominance*) Given two solutions, namely x and $y \in R^n$, we say that $x \leq y$ if $x_i \leq y_i$ for $i = 1, 2, \ldots, n$ and that x dominates by y if $x \leq y$ and $x \neq y$.
2. **Definition 2** (*Non-dominance*) A vector of decision variables $x \in X \subset R^n$ is non-dominated with respect to X, if no $y \in X$ exist, such that $f(y) \leq f(x)$.
3. **Definition 3** (*Pareto optimality*) A vector of decision variables $x^* \in X \subset R^n$ (F is the feasible region) is Pareto optimal if it is non-dominated with respect to F.
4. **Definition 4** (*Pareto optimal set*) The Pareto optimal set P^* is defined as follows: P^*={ $x \in F$, such that, x is Pareto optimal}.
5. **Definition 5** (*Pareto front*) The Pareto front PF^* is defined by the following: PF^*={ $f(x) \in R^k; x \in P^*$}

2.1 Multi-objective gravitational search algorithm

Gravitational search algorithm is a new algorithmic method that uses Newton's laws of motion and gravity in different modification techniques [24]. Nonetheless, in research, the algorithms have been developed because little is known about its inception. In *GSA*, for each mass (agent), one can consider four specifications: inertial mass, position, active gravitational mass and passive gravitational mass. Such as:

$$m_i(t) = [f_i t_i(t) - \text{worst}(t)]/[\text{best}(t) - \text{worst}(t)], \quad i = 1, 2, 3, \ldots, n_{\text{mass}} \tag{2}$$

$$M_i(t) = m_i(t)/\sum_{j=1}^{N} m_j(t), \quad 0 \leq M_i(t) \leq 1 \tag{3}$$

where $f_i t_i$ represents the objective function of the agent ith; the worst value worst(t) represents the lowest value of the objective function (for a minimisation problem); t represents the time; and N represents the number of the objective functions or the size of the swarm Eq. 2. To calculate the acceleration of the agent, the sum of the forces of heavy masses applied should be considered on the basis of the law of gravity and the acceleration of the agent using the motion law in Eq. 3. The current velocity is added to the acceleration using Eq. 5. Afterwards, the position is calculated by using Eq. 4.

$$a_i(t) = G(t) \sum_{j=1}^{n_{\text{mass}}} r_j \times [(M_j(t)/R_{i,j} + \varepsilon] \times \left(x_j^d(t) - x_i^d(t)\right) \tag{4}$$

Thus, the following equations are obtained:

$$v_i^d(t+1) = r_i \times v_i^d(t) \times (a_i^t) \tag{5}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{6}$$

In Eq. (4), $G(t)$ represents a constant of gravity and the ε value is considerably small. $R_{i,j}$ is the Euclidean distance between two agents, namely i and j. r_i and r_j represent two random numbers between 0 and 1 that ensure the random properties of the algorithm.

MOPs are observed, and a single fitness function must correspond to each solution. Given that the use of an equivalent mass for a population that contains an objective function is not possible, we used the SPEA II method to determine the equivalent of the mass that is dependent on the MOGSA algorithm. To avoid being controlled by individuals, wherein some exhibit the same objective function in the same archive, we considered the dominant solutions and the controlled solutions with SPEA II. Each individual i in the population P_t displays strength $S(i)$ as in Eq. 7, which represents the number of controlled solutions

$$S(i) = \mid \{j : j \in x_t + \bar{x}_t \wedge i \succ j\} \mid, \tag{7}$$

where (|.|) signifies the basic character of the set, + denotes the union over a set, and the symbol ($\succ$) denotes the relationship to the dominant Pareto solution. Based on the values of $S(i)$, the raw objective function R_i is calculated in each individual agency using Eq. 8 as follows:

$$R_i = \sum_{j \in x_t + \bar{x}_t \wedge i \prec j} S_j \tag{8}$$

This simple function R_i is calculated through the strengths of archive and population. On the contrary, in *SPEAII*, the archive members are in control of the situation. Notably, the objective functions should be in the form of miniaturisation, that is $R_i = 0$, corresponding to a non-dominated individual, whereas its highest R_i value indicates that i is controlled by many individuals, which consequently dominates many individuals.

Although the designation of the explicit function provides a type of mechanism (niching mechanism) that is

based on the concept of Pareto dominance, it may fail when most individuals do not dominate one another. Additional density information is added between individuals who exhibit objective functions with identical values. The density estimation technique is an intensified method (nearest neighbourhood method) [25], in which the density at any point is a decreasing feature of the distance to the kth nearest neighbour.

The present study considers the inverse distance (the nearest k-th) as an estimate of intensification. Particularly, for any individual i, the spaces between the individual j in the archive and the community are calculated and stored as a list. After the sorting in ascending order, the kth element provides the required distance, which is represented by Q_i^k.

The present study uses k, which is equal to the square root of sample size [25]; thus, $k = \sqrt{(N + \bar{N})}$. Subsequently, density D_i, which corresponds to i, can be defined as follows:

$$D_i = 1/Q_i^k + 2 \tag{9}$$

Number 2 is added to the denominator to ensure that its value is higher than 0, and $D_i \prec 1$. Finally, the number is added to the explicit D_i function of the initial individual i to obtain the new fitness F_i as shown in Eq. 10:

$$F_i = R_i + D_i \tag{10}$$

In Eq. 11, the researcher used the function F_i in a new equation as an exponential function by using the technique applied in the solution for *SPEAII* algorithm.

$$m_{i(t)} = \exp\left(-F(i)\right) \tag{11}$$

input : Set $k = 0$, velocity=0
Randomly initialize point X_i for n population
Calculate the fitness values of initial population: $f(X)$;
Find the non-dominated solutions and initialized the archive with them
output: Non-dominated solutions set
while *(the termination conditions are not met)* **do**
  MOGSA Steps;
  Calculate the mass function m
  Calculation of Gravitational constant G
  Calculation of acceleration in gravitational field a
  $v_{t+1} = randv_i + a$
  $x_{t+1} = x_t + v_{t+1}$
  Find the non-dominated solutions $\bar{x}_{t+1}$
  Copy all non-dominated individuals in x_t and $\bar{x}_t$ to $\bar{x}_{t+1}$
  if *size of* $\bar{x}_{t+1}$ *exceeds* $\bar{N}$ *then reduce* $\bar{x}_{t+1}$ **then**
    $k = 1$
    while *min* (Q_i^k)=*max* $(Q_i^{k_1})$ *and* $k <$ *size* Q_i **do**
      $k = k + 1$;
    end
    $j = min(Q_i^k)$
    $x_{t+1}^k = null$
    $Q_i^k = null$
  end
  otherwise if size of $\bar{x}_{t+1}$ is less than $\bar{N}$ then fill $\bar{x}_{t+1}$ with dominated individuals in X_t and $\bar{x}_{t+1}$.
  Set $k = k + 1$;
end

Algorithm 1: *MOGSA* Procedure

2.2 Multi-objective bat algorithm

Bats are mammals with wings and echolocation ability. Approximately 996 different bat species have been identified worldwide, and they account for approximately 20% of all mammal species [30]. In Ref. [4], a new optimisation algorithm known as BAT is proposed on the basis of swarm intelligence and bat observation. One can simulate the parts of the echolocation characteristics of microbat by using the BAT. The advantages of this algorithm include simplicity, flexibility and easy implementation. Furthermore, the algorithm efficiently solves a wide range of problems, such as highly nonlinear ones [10]. BAT also provides promising optimal solutions quickly and works well with complicated problems. The disadvantages of this algorithm are convergence occurring quickly at early stages and the decrease in convergence. In addition, no mathematical analysis links the parameters with convergence rates. The most suitable values for most applications are also unclear [20].

Equations 12–14 simulate the movement of bat virtual agencies:

$$Q_i = Q_{\min} + (Q_{\max} - Q_{\min})\beta^* \tag{12}$$

$$v_i^t = v_i^{t-1} + (P_i^{t-1} - P_{\text{best}})f_i \tag{13}$$

$$P_i^t = P_i^{t-1} + v_i^t \tag{14}$$

where Q is the frequency used by bats to obtain prey; $Q_{\max}$ and $Q_{\min}$ represent the minimum and the maximum limits, respectively; P_i represents the location of the ith bat in the search space; $v_i(t)$ represents the velocity of the bat; and t indicates the number of iterations. In addition, β has a range of [0, 1], and it is plotted through a uniform distribution. P_{best} represents the most suitable solution found for all the populations.

To obtain better optimal procedure for multi-objective functions using BAT, we develop an algorithm called MOBAT [28] by introducing two new components (i.e. archive and leader), as found in the MOPSO algorithm in Ref. [3]. The archive is responsible for saving and restoring the most remarkable non-dominated and non-controllable Pareto optimal solutions that have been obtained to date. The archive also displays a main unit, which is the control unit of the archive. This unit controls the number of non-controlling solutions when new non-controlling solutions exist. Concurrently, the archive size is complete. During the process of replication, the non-dominated solutions obtained against the archive population are compared. Consequently, four different situations will be observed.

1. The new member is logged into the archive if a member of the archive is in control, in which the user is allowed access to the archive.

2. The new solution dominates the solution of one or more of the solutions in the archive. In this case, the solution or the dominant solutions in the archive must be deleted. The new solution will be able to access the archive.
3. If neither the new solution nor the archive member dominates each other, then a new archive solution must be added.
4. If the archive is full, then the network mechanism is run first to repartition the target space, determine the busiest sector and delete one of the existing solutions. The new solution should subsequently be incorporated into a less crowded slot in the system to improve the final diversification of Pareto approximate solution.

Increasing the probability of deleting a solution is proportional to solutions in a hypercube (segment). A special case exists in which a solution is inserted by hypercube. In this case, all sectors are extended to cover new solutions. Therefore, other solutions can also be changed. The second mechanism is selecting a leader (where the leader directs the selected members within the research area). In MOBAT algorithm, the most suitable obtained solution is used. This leader directs members within the research area to obtain a solution near the most suitable solution.

However, solutions cannot be in a multi-objective search space compared with the ideal Pareto concepts. The leader selection mechanism is designed to handle the issue. An archive contains the most suitable non-dominant solutions obtained. The leader selects the component from the crowded segments of the space solution and offers one of the non-dominant solutions. Selection is performed through a roulette wheel with the following possibility for each hypercube:

$$P_i = c/N_i \tag{15}$$

where c is a constant number higher than 1, and N_i is the number of obtained Pareto optimal solutions in the ith segment. The equation indicates that the lack of congestion in the hypercube shows a high probability in the proposal of a new leader.

input : Set $k = 0$, velocity=0,$\mu = 0.1, r^0 = 0.5, A = 0.6$
Randomly initialize point P_i for n population
Calculate the fitness values of initial population: $f(P)$;
Find the non-dominated solutions and initialized the archive with them
output: Non-dominated solutions set

while *(the termination conditions are not met)* **do**
  MOBAT Steps;
  $Q = Q_{min} + (Q_{max} - Q_{min}) * rand$
  $P_{leader1}$ = Select Leader (archive)
  $v_{t+1} = v_i + (P_{Leader1} - P_t) * Q$
  $P_{new} = P_t + v_{t+1}$
  if $rand > r$ **then**
    $P_{leader1}$ = Select Leader (archive)
    $P_{new} = P_i + rand * (P_{Leader2} - P_t)$
  end
  if P_{new} *dominated on* P_t *and* $(rand < A)$ **then**
    $P_t = P_{new}$
  end
  if $rand < ([1 - (k-1)/maxiteration - 1])^{1/\mu}$ **then**
    S = Mutation (P_t)
    if P_{new} *dominated on* P_t *and* $(rand < A)$ **then**
      $P_t = S$
    end
  end
  Find the non-dominated solutions
  Update the archive with respect to the obtained non-dominated solutions
  if *the archive is full* **then**
    Run the grid mechanism to omit one of the current archive members
    Add the new solution to the archive
  end
  if *any of the new added solutions to the archive is located outside the hyper cubes* **then**
    Update the grids to cover the new solution(s)
  end
  Increase r and reduce A
  Set $k = k + 1$;
end

Algorithm 2: $MOBAT$ Procedure

In this case, the possibility of selecting a hypercube to choose a driver is increased when the number of solutions obtained in hypercube decreases. The following procedure shows the action of the BAT for multi-objective function and in solving MOPs by using the mechanical Pareto optimal solution from the optimal solution.

2.3 Multi-objective gravitational search algorithm with bat algorithm

This section discusses the optimal hybridisation algorithm by means of a communication strategy between the two algorithms [1]. The idea is based on three stages. The first stage is based on creating a MOGSA from SPEA II theory as follows. We benefit from the theoretical work found in the SPEA II algorithm concerning the selection of the most suitable solution but not for multiple objective functions, because the work of this algorithm depends on more than one objective function. This approach benefits from the equation that describes the sum of non-dominated solutions and the distance between two individuals in the population.

The function F_i is the criterion of differentiation between them and the mass of each individual in population because the proportionality between them and the mass of the opposite suits any possibility when the low F_i

increases the mass of any solution well, and vice versa. Remarkably, *MOGSA* can work if $(\beta < 0.5)$, where the parameter (β) is optional. Consequently, we can distinguish or decide which algorithm works first in the multi-objective space.

The second stage is based on creating a multi-objective BAT from MOPSO theory as follows. We benefit from the theoretical work found in MOPSO with regard to the selection of the most suitable solution but not for multiple objective functions, because the work of this algorithm depends on more than one objective function. This algorithm contains two important properties in selecting the individual with the most suitable solution. These two features are leader and archive, where the leader of the swarm is selected from the internal archive located in the solution space. The leader is the most remarkable solution for the *MOBAT* algorithm. Furthermore, the algorithm is based on the same parameter used by the *MOGSA* algorithm. If the parameter (β) is higher than 0.5, then the algorithm is suitable. Moreover, the random input population generated from the MOGSA algorithm is considered the output of the MOBAT algorithm for the previous iteration and vice versa; this update works in each iteration.

input : Set $k = 0$, velocity=0,$\beta = 0.7, r^0 = 0.5, A = 0.6$
Randomly initialize point P_i for n population
Calculate the fitness values of initial Population: $f(P)$;
Find the non-dominated solutions and initialized the archive with them
output: Non-dominated solutions set
while *(the termination conditions are not met)* **do**
  if $rand < \beta$ **then**
    MOGSA Steps;
    Calculate the mass function m
    Calculation of Gravitational constant G
    Calculation of acceleration in gravitational field a
    $v_{t+1} = randv_i + a$
    $x_{t+1} = x_t + v_{t+1}$
  end
  MOBAT Steps;
  $Q = Q_{min} + (Q_{max} - Q_{min}) * rand$
  $P_{leader1}$ = Select Leader (archive)
  $v_{t+1} = v_i + (P_{Leader1} - P_t)Q$
  $P_{new} = P_t + v_{t+1}$
  if $rand > r$ **then**
    $P_{leader2}$ = Select Leader (archive)
    $P_{new} = P_i + rand * (P_{Leader2} - P_t)$
  end
  if P_{new} *dominated on* P_t *and* $(rand < A)$ **then**
    $P_t = P_{new}$
  end
  if $rand < ([1 - (k-1)/maxiteration - 1])^{1/\mu}$ **then**
    S = Mutation (P_t)
    if P_{new} *dominated on* P_t *and* $(rand < A)$ **then**
      $P_t = S$
    end
  end
  Update all solutions in population Step 3
  for *each point i in the population* **do**
    $c = rand * int(1, 2)$
    $P_{leader3}$ = Select Leader (archive)
    $P_{new} = P_i + rand * (P_{Leader3} - c * \bar{P})$
    if P_{new} *dominated on* P_t *and* $(rand < A)$ **then**
      $P_t = S$
    end
  end
  Find the non-dominated solutions
  Update the archive with respect to the obtained non-dominated solutions
  if *the archive is full* **then**
    Run the grid mechanism to omit one of the current archive members
    Add the new solution to the archive
  end
  if *any of the new added solutions to the archive is located outside the hyper cubes* **then**
    Update the grids to cover the new solution(s)
  end
  Increase r and reduce A
  Set $k = k + 1$;
end

Algorithm 3: *MOGSABAT* Procedure

The final stage is the creation of the two algorithms to obtain a hybrid algorithm, which is our objective. The following two algorithms are run, and the hybrid algorithm will perform an update on the new solutions. The update also depends on the most remarkable solution. We then work on these solutions in the same manner as the MOBAT algorithm [31]. We obtain a population from both algorithms. The present study needs to archive all non-dominated solutions and restart this work until an improved solution that is near the most suitable solution is obtained. These stages can be observed clearly from the procedure.

3 Evaluation methodology, results and discussion

This section evaluates the performance of the proposed MOGSABAT algorithm. Firstly, we describe the evaluation methodology and present the results of the experiments, which are conducted in one scenario with different optimisation problems. Secondly, we compare the performance of the MOGSABAT algorithm with that of other intelligent computational techniques. Thirdly, we discuss our findings in detail.

3.1 Evaluation methodology

The evaluation methodology employed in this work is divided into three parts. The first part describes the benchmarking of the optimisation problem (tri and bi) for evaluating the performance of the proposed MOGSABAT algorithm. The second part compares the performance of the MOGSABAT algorithm with that of other intelligent computation algorithms. The third part explains the procedure of the proposed MOGSABAT algorithm.

3.1.1 Benchmark of the *Tri* and *Bi* optimisation problem

According to no free lunch (NFL) theorem, for any algorithm, any elevated performance over one class of problems is exactly paid for in performance over another class [13, 32]. A particular metaheuristic may yield promising results for a set of problems but may perform poorly on another set of problems. With NFL, this field of study is highly active. As a consequence, the extant approaches are enhanced, and new metaheuristics are being proposed every year.

The details formulations of nine multi-objective test problem with bias feature as are follows: $BT1$ this problem has two objective functions and 30 variables. In the function $BT2$, there are same variables as that in the function $BT1$ and $x \in [0, 1]^{30}$.

All the functions from $BT3 - BT9$ are the same in the details; only the function $BT7$ is different in the space area, that is, $x \in [0,1] * [-1,1]^{29}$.

In this subsection, we use two groups of benchmark optimisation problem (i.e. UF and ZDT); these two groups of function are illustrated in detail in [32]. In addition, we test these functions with iteration $i = 20$ and populations $n = 100$.

3.2 Performance-based metrics for multiple-objective

1. Generational Distance GD In determining whether solutions of Q can be included with the set of P^* or not, the use of the (GD) metric is appropriate for it [18] because it estimates the average distances of the solution sets of Q from P^* as follows:

$$\mathrm{GD} = \sum_{i=1}^{Q} (d_i^p)^{1/p}/Q \tag{16}$$

for two objective functions ($p = 2$) where p represents objective function. The parametric value d_i denotes Euclidean distances (in the objective spaces) across the solution $i \in Q$ to the most proximate members P^*, which can be obtained as follows:

$$d_i = \min_{k \in P^*} \sqrt{\sum_{m=1}^{M} \left(f_m^i - f_m^*(i)\right)^2} \tag{17}$$

where $f_m^*(i)$denotes the ith objective function of the kth member of P^*. Algorithm methods with lower values of GD are intuitively more reliable.

2. Reserve Generational Distance RGD

$$\mathrm{RGD} = \sum_{i=1}^{Q} (d_i^p)^{1/p}/Q \tag{18}$$

where Q represents criminality assigned to the P^* set, which is also known as the RGD metric.

3.2.1 Evaluation procedure

The experimental results are described based on the mean, SD and Wilcoxon signed-rank test statistic of the function values.

4 Statistical measurement

1. **Mean**: Mean ($\bar{x}$) is computed as the sum of all the observed outcomes from the sample divided by the total number of these outcomes as shown as follows:

$$\bar{x} = 1/n \sum_{i=1}^{n} x_i \tag{19}$$

2. **Standard Deviation (SD)**: SD quantifies the variation or dispersion of a set of data for the function values as shown as follows:

$$\mathrm{SD} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2} \tag{20}$$

3. **Wilcoxon Signed-Rank Test**: The Wilcoxon signed-rank test determines the difference between two samples [6] and provides an alternative test of location that is affected by the magnitudes and signs of these differences. This test also checks whether one algorithm outperforms the other. Let d_i denote the difference between the performance scores of two algorithms in solving ith out of n problems. Let R^+ denote the sum of ranks for the problems in which the first algorithm outperforms the second (Eq. 21), and let R^- represent the sum of ranks for the problems in which the second algorithm outperforms the first (Eq. 22). The ranks of $d_i = 0$ are split evenly among the sums. If these sums have an odd number, then one of them is ignored.

$$R^+ = \sum_{d_i \succ 0} \mathrm{rank}(d_i) + 1/2 \sum_{d_i = 0} \mathrm{rank}(d_i) \tag{21}$$

$$R^- = \sum_{d_i \prec 0} \mathrm{rank}(d_i) + 1/2 \sum_{d_i = 0} \mathrm{rank}(d_i) \tag{22}$$

We use *MATLAB* to find the p value for comparing the algorithms at a significant level of $\alpha = 0.05$. The null hypothesis is rejected when the p value is less than the significant level. R^+ represents a high mean algorithm that shows superiority over other algorithms across different sets of experiments. When $R^+ = n \times (n-1)/2$, this algorithm outperforms all algorithms across all experiments.

4.1 Results

The results of the experiments are described based on one scenario. The proposed MOGSABAT algorithm and the benchmarking functions that are developed based on three groups of unconstrained optimisation problems in Sect. 3.1.1 are compared with the intelligent computation techniques described in Sect. 3.2. The mean, SD and Wilcoxon signed-rank test statistic of algorithms are discussed in Sect. 4.

Each benchmark function has 30 runs, and each algorithm has 100 iterations and 30 variables, as shown below form Tables 1, 2, 3, 4, 5, 6. Each compared algorithm

independently performs 30 runs, and the symbols +, = and − denote whether the GD and RGD results of the proposed MOGSABAT are statistically better than, equal to or worse than those of the corresponding peer competitors with a significant level 0.05, respectively.

4.2 Discussion

Table 1 compares the algorithms and instances depending on *GD*, *mean* and *SD* criteria. Results show that the excellent performance of the hybrid *MOGSABAT* algorithm is observed in three instances of multiple functions: *ZDT*1, *ZDT*2 and *ZDT*3, whereas *SPEAII* and *MOGSA* are the most suitable algorithms in *ZDT*4 and *ZDT*6 functions, respectively.

In Fig. 1, the calculation and comparison among the algorithms are shown to determine which one is the best. Using the *Wilcoxon* rank sum test for the performance *GD*, for functions *ZDT*1 and *ZDT*4, the proposed *MOGSABAT* algorithm is better than the *MOGSA* algorithm. For functions *ZDT*1 and *ZDT*2, the control algorithm clearly exceeds the *MOBAT*, *MOPSO* and multi-objective non-sorting genetic algorithm (*MONSGAII*) algorithms [16]; however, in *ZDT*4, the *MOGSABAT* is outperformed by the *SPEAII* algorithm.

Table 2 compares the algorithms and instances that depend on the *RGD*, *mean* and *SD* criteria. Results indicate the excellent performance of *MOGSABAT* in three instances of multiple functions *ZDT*1, *ZDT*2 and *ZDT*3 and *ZDT*6. For *ZDT*4, *SPEAII* displays the most remarkable mean and *SD* values.

Table 1 GD results of MOGSABAT against MOGSA, MOBAT, MPSO, NSGA II and SPEA II over ZDT1–ZDT6 test problems

Problem	MOGSABAT	MOGSA	MOBAT	MPSO	NSGAII	SPEAII
ZDT1	0.01(0)	3.40(5.81)+	63.94(9.39)+	15.80(12.51)+	44.58(4.36)+	4.42(1.38)+
ZDT2	0(0)	0.90(2.76)−	69.42(9.91)+	28.42(15.31)+	47.44(4.79)+	5.36(1.08)+
ZDT3	0.02(0.01)	4.43(8.95)−	65.09(13.59)+	22.55(15.28)=	42.92(4.267)=	5.22(1.56)+
ZDT4	12.44(6.34)	125.87(20.99)+	36.72(17.37)+	36.40(15.57)+	16.52(5.41)+	4.29(2.30)−
ZDT6	0.95(0.87)	0.05(0.20)−	9.18(1.23)+	1.40(2.08)+	604.45(1113.49)+	2.08(1.02)+
+/=/−		2/=/3	5/=/−	4/1/−	4/1/−	4/=/1

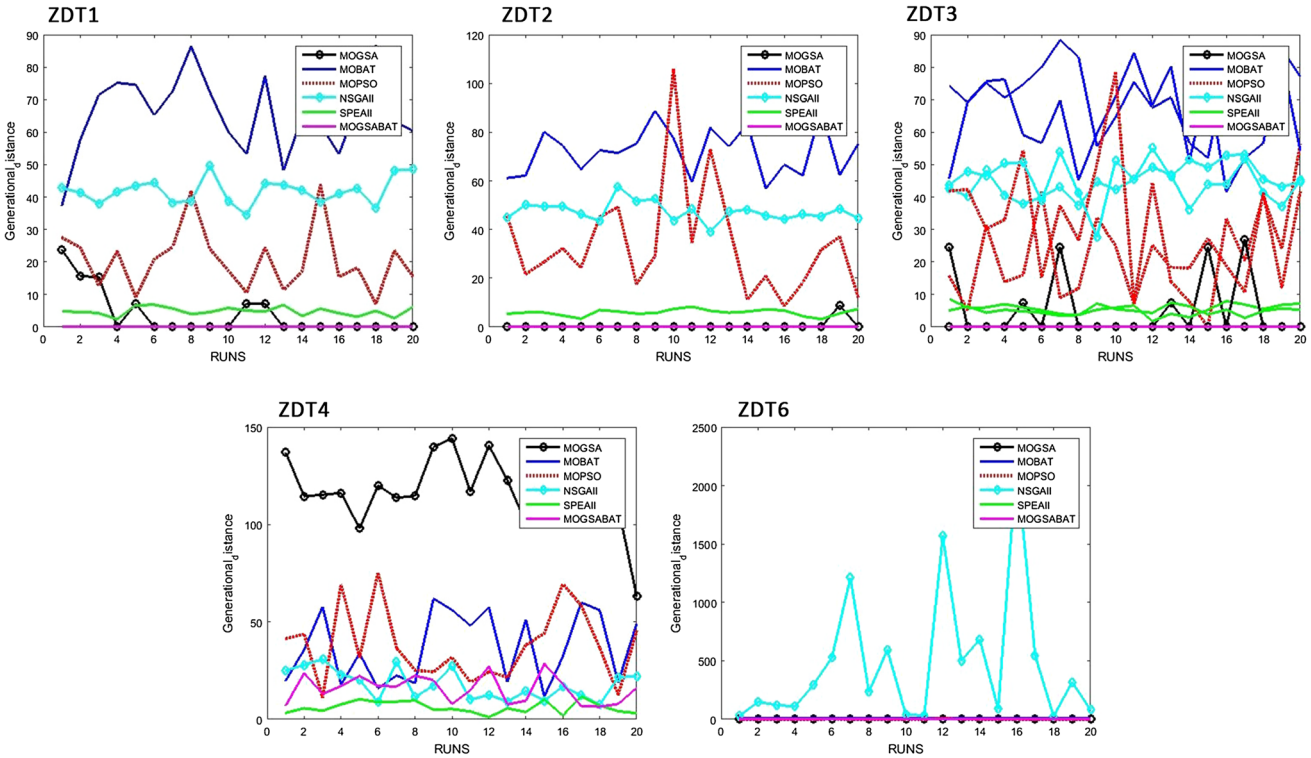

Fig. 1 Visualisation of GD performance metric by MOGSABAT against the five competitive algorithms in ZDT

Table 2 RGD results of MOGSABAT against MOGSA, MOBAT, MPSO, NSGA II and SPEA II over ZDT1–ZDT6

Problem	MOGSABAT	MOGSAy	MOBAT	MPSO	NSGAII	SPEAII
ZDT1	0.18(0.34)	3.68(5.04)+	64.04(9.14)+	16.13(12.64)+	40.92(4.27)+	4.61(1.31)+
ZDT2	0.177(0.27)	1.44(2.67)+	69.75(9.90)+	28.78(15.28)+	43.75(4.34)+	5.63(1.12)+
ZDT3	0.25(0.33)	4.46(7.75)+	64.13(12.99)+	22.91(15.51)+	38.90(3.91)+	5.42(1.42)+
ZDT4	12.72(5.64)	124.30(21.87)+	34.71(16.49)+	33.24(13.19)+	6.16(1.73)−	4.31(1.78)−
ZDT6	0.04(0.08)	0.10(0.22)−	9.36(1.24)+	0.46(1.85)+	500.15(1061.78)+	0.68(0.98)+
+/=/−		2/=/3	5/=/−	5/=/−	4/=/1	4/=/1

In Fig. 2, the calculation and comparison among the algorithms are demonstrated to determine which one is the best. Using *RGD* performance, for functions *ZDT*1 and *ZDT*4, the proposed *MOGSABAT* algorithm is better than the *MOGSA* algorithm. For the functions *ZDT*1 and *ZDT*6, the control algorithm clearly exceeds the *MOBAT* and *NSGAII* algorithms; however, in the function *ZDT*4, the *MOGSABAT* is outperformed by the *SPEAII* algorithm.

Table 3 compares the algorithms and instances depending on the *GD*, *mean* and *SD*. Results demonstrate the superiority of *NSGAII* [5] in three instances of multiple functions *UF*1, *UF*2 and *UF*3. This algorithm also shows a good *SD* result for instance *UF*5. In instance *UF*4, the *MOPSO* algorithm obtains the most remarkable mean and *SD* values. *SPEAII* also shows the most desirable *SD* value in instance *UF*5, and *NSGAII* displays the most remarkable *SD* value in instance *UF*6. In addition, *MOGSABAT* shows superiority in terms of *mean* and *SD* values in instance *UF*7.

As shown in Fig. 3, the proposed *MOGSABAT* algorithm has better *GD* performance in comparison with *MOGSA* and *MOBAT* algorithms for functions *UF*1 to *UF*7 if we take $a = 0.001$ (where a represents the statistical level). Although the proposed algorithm performs equally with the *NSGAII* algorithm in function *UF*1, it is better in functions *UF*2, *UF*4, *UF*5 and *UF*6. Finally, the hybrid algorithm is better compared with the *SPEAII* algorithm in functions *UF*1, *UF*4, *UF*5 and *UF*6.

Table 4 compares the algorithms and instances that depend on the *RGD*, *mean* and *SD* criteria. Results show that *NSGAII* displays the most remarkable *SD* value in the instances of multiple functions *UF*2, *UF*3, *UF*5 and *UF*6

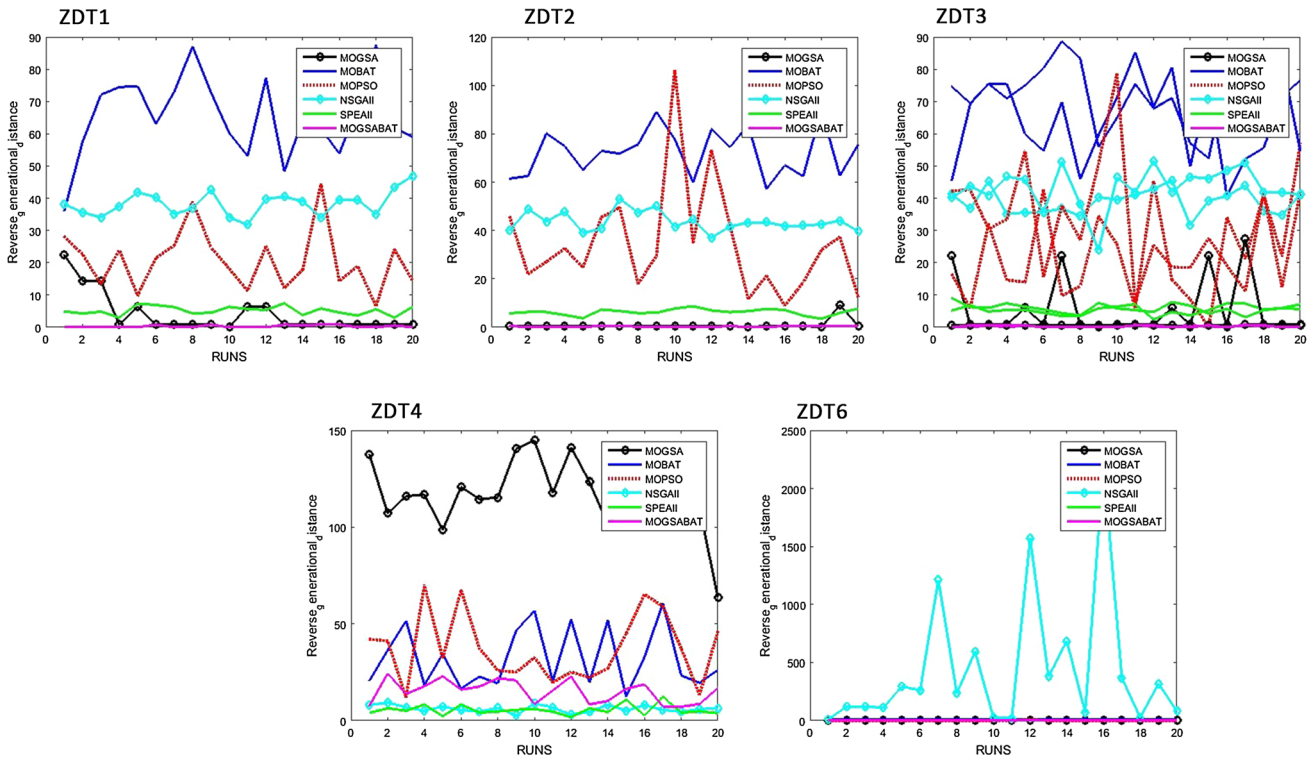

Fig. 2 Visualisation of RGD performance metric by MOGSABAT against the five competitive algorithms in ZDT

Table 3 GD results of MOGSABAT against MOGSA, MOBAT, MPSO, NSGA II and SPEA II over UF1–UF7

Problem	MOGSABAT	MOGSA	MOBAT	MPSO	NSGAII	SPEAII
UF1	0.294(0.116)	0.357(0.227)+	0.782(0.153)+	0.823(0.253)−	0.225(0.054)=	0.246(0.124)+
UF2	0.177(0.075)	0.714(0.162)+	0.155(0.034)+	0.500(0.104)+	0.141(0.050)+	0.076(0.018)−
UF3	0.401(0.080)	1.105(0.911)+	0.609(0.074)+	1.024(0.357)+	0.200(0.065)−	0.380(0.030)−
UF4	0.135(0.011)	0.132(0.014)+	0.170(0.009)+	0.129(0.010)=	0.429(0.187)+	0.134(0.009)+
UF5	1.882(0.573)	2.686(0.760)+	4.013(0.716)+	3.766(1.212)+	2.087(0.406)+	2.254(0.411)+
UF6	1.882(0.573)	2.686(0.760)+	4.013(0.716)+	3.766(1.212)+	2.087(0.406)+	2.254(0.411)+
UF7	0.359(0.204)	0.343(0.193)+	0.867(0.273)+	0.814(0.385)+	0.222(0.0767)−	0.138(0.053)−
+/=/−		7/=/−	7/=/−	5/1/2	4/1/2	4/=/3

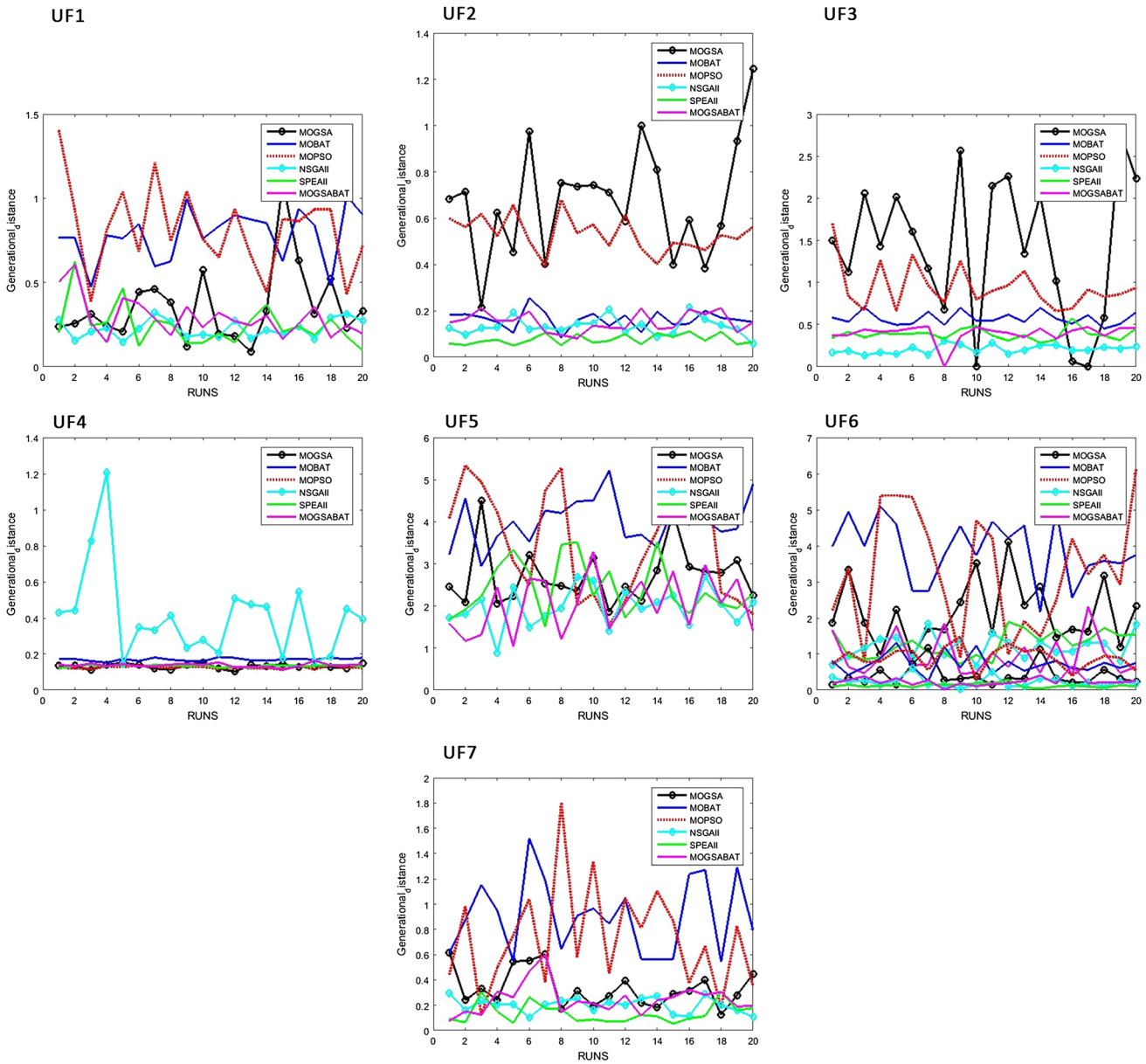

Fig. 3 Visualisation of GD performance metric by MOGSABAT against the five competitive algorithms in UF

Table 4 RGD results of MOGSABAT against MOGSA, MOBAT, MPSO, NSGA II and SPEA II over UF1–UF7

Problem	MOGSABAT	MOGSA	MOBAT	MPSO	NSGAII	SPEAII
UF1	0.219(0.061)	0.408(0.203)+	0.727(0.149)+	0.643(0.196)+	0.239(0.034)+	0.227(0.053)+
UF2	0.120(0.015)	0.787(0.118)+	0.193(0.025)+	0.414(0.055)+	0.095(0.024)−	0.104(0.007)−
UF3	0.446(0.052)	1.298(0.558)+	0.662(0.067)+	0.803(0.096)+	0.341(0.021)−	0.445(0.030)+
UF4	0.123(0.012)	0.119(0.009)+	0.159(0.010)+	0.117(0.010)=	0.141(0.017)+	0.119(0.008)+
UF5	1.646(0.382)	2.636(0.710)+	3.780(0.603)+	3.291(0.931)+	1.856(0.307)+	2.108(0.358)+
UF6	0.875(0.316)	2.590(1.578)+	3.305(0.543)+	1.964(0.895)+	1.005(0.227)+	1.191(0.272)+
UF7	0.295(0.094)	0.406(0.182)+	0.862(0.175)+	0.642(0.220)+	0.413(0.074)+	0.166(0.053)−
+/=/−		7/=/−	7/=/−	6/=/1	5/=/2	5/=/2

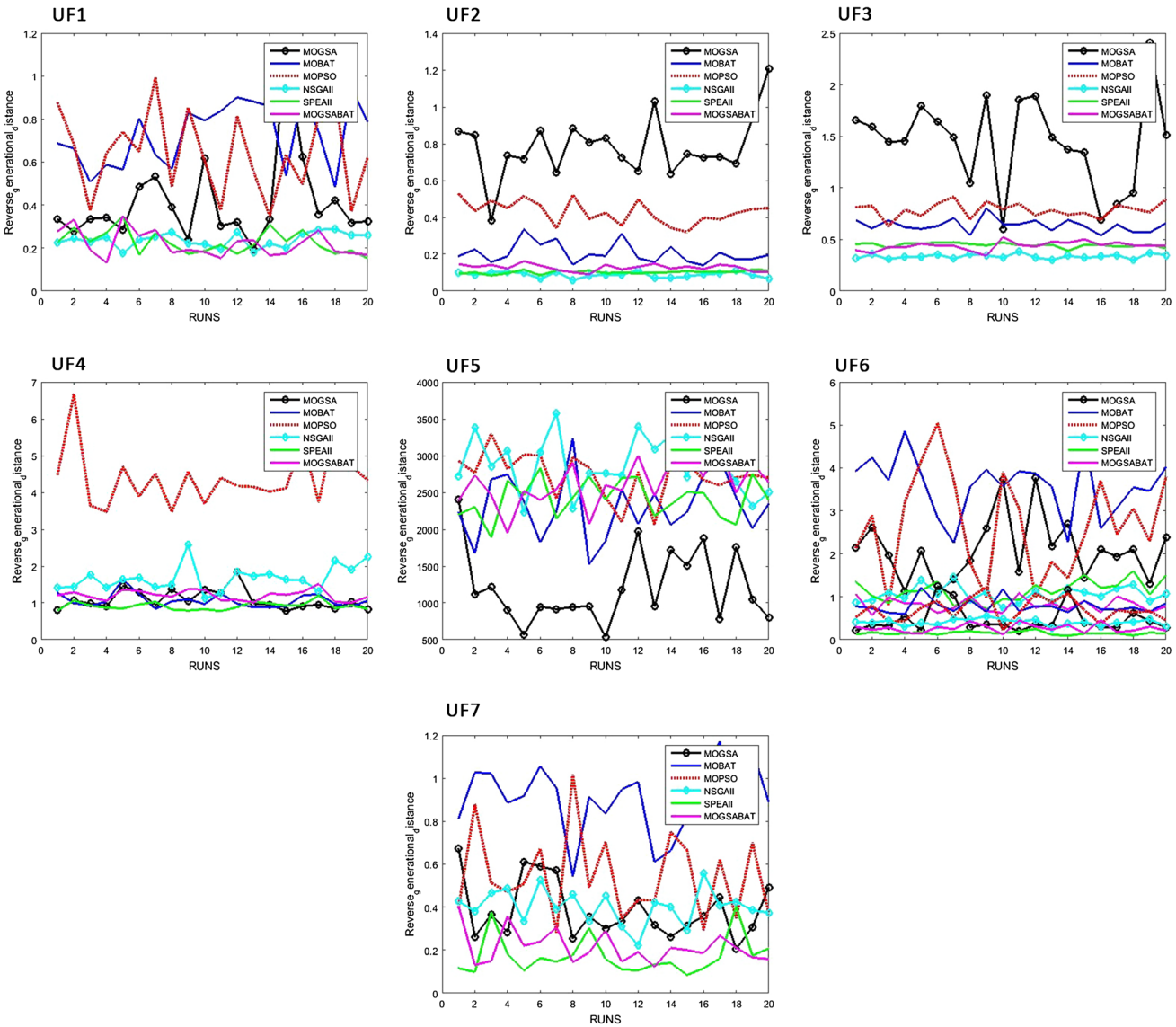

Fig. 4 Visualisation of RGD performance metric by MOGSABAT against the five competitive algorithms in UF

Table 5 GD results of MOGSABAT against MOGSA, MOBAT, MPSO, NSGA II and SPEA II over BT1–BT9

Problem	MOGSABAT	MOGSA	MOBAT	MPSO	NSGAII	SPEAII
BT1	4.071(0.134)	4.507(0.193)+	4.770(0.235)+	4.518(0.235)+	4.099(0.040)+	3.906(0.097)−
BT2	0.879(0.040)	1.054(0.068)+	0.934(0.038)+	0.874(0.037)+	0.882(0.034)+	0.886(0.025)+
BT3	0.981(0.030)	1.072(0.027)+	1.002(0.016)+	0.971(0.040)+	0.994(0.013)+	1.046(0.040)−
BT4	4.018(0.130)	4.397(0.288)+	4.906(0.285)+	4.419(0.225)+	4.088(0.10)+	3.790(0.092)−
BT5	3.993(0.229)	4.402(0.261)+	4.721(0.221)+	4.296(0.166)+	4.031(0.044)+	3.865(0.077)+
BT6	0.007(0.008)	1.080(0.977)+	3.742(0.349)+	4.629(0.473)+	1.514(0.328)+	1.957(0.258)+
BT7	2.098(0.999)	8.555(3.720)+	5.839(1.018)+	6.658(1.517)+	0.701(0.230)−	1.695(0.354)+
BT8	0.222(0.261)	4.669(7.114)+	9.392(1.623)+	14.405(3.389)+	fail(fail)+	5.014(0.411)+
BT9	3.339(0.112)	3.247(0.159)=	3.571(0.139)+	3.840(0.179)+	3.475(0.078)+	3.141(0.038)−
+/=/−		8/1/−	9/=/−	9/=/−	8/=/1	5/=/4

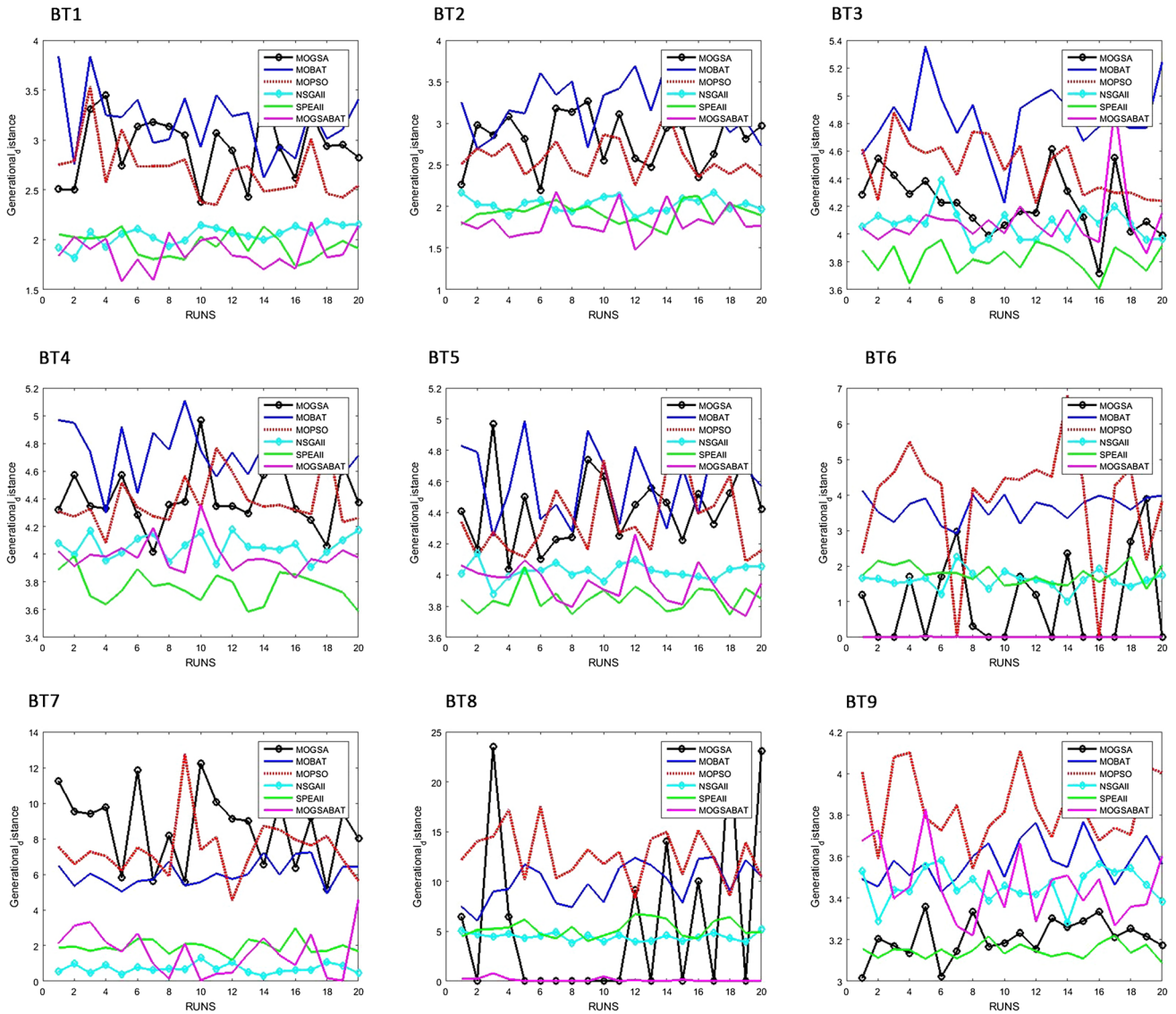

Fig. 5 Visualisation of GD performance metric by MOGSABAT against the five competitive algorithms in BT

Table 6 RGD results of MOGSABAT against MOGSA, MOBAT, MPSO, NSGA II and SPEA II over BT1–BT9

Problem	MOGSABAT	MOGSA	MOBAT	MPSO	NSGAII	SPEAII
BT1	4.038(0.099)	4.501(0.156)+	4.793(0.242)+	4.388(0.202)+	4.122(0.054)+	3.912(0.091)−
BT2	1.759(0.177)	2.689(0.388)+	3.280(0.327)+	2.732(0.283)+	2.047(0.092)+	1.997(0.105)+
BT3	4.082(0.101)	4.219(0.300)+	4.899(0.284)+	4.536(0.226)+	4.152(0.085)+	3.808(0.115)−
BT4	3.970(0.110)	4.426(0.258)+	4.870(0.283)+	4.333(0.168)+	4.050(0.083)+	3.769(0.103)−
BT5	3.952(0.118)	4.488(0.251)+	4.780(0.217)+	4.274(0.144)+	4.095(0.044)+	3.896(0.078)+
BT6	0.827(0.012)	1.603(0.749)+	3.830(0.340)+	4.449(0.369)+	1.443(0.275)+	2.017(0.231)+
BT7	1.992(0.779)	8.359(3.335)+	5.797(0.891)+	6.201(1.063)+	0.669(0.189)−	1.773(0.326)+
BT8	0.695(0.070)	5.087(6.313)+	9.242(1.447)+	12.822(2.786)+	fail(fail)+	4.862(0.365)+
BT9	3.188(0.086)	3.232(0.164)+	3.514(0.127)+	3.453(0.106)+	3.267(0.133)+	3.098(0.051)−
+/=/−		9/=/−	9/=/−	9/=/−	8/=/1	5/=/4

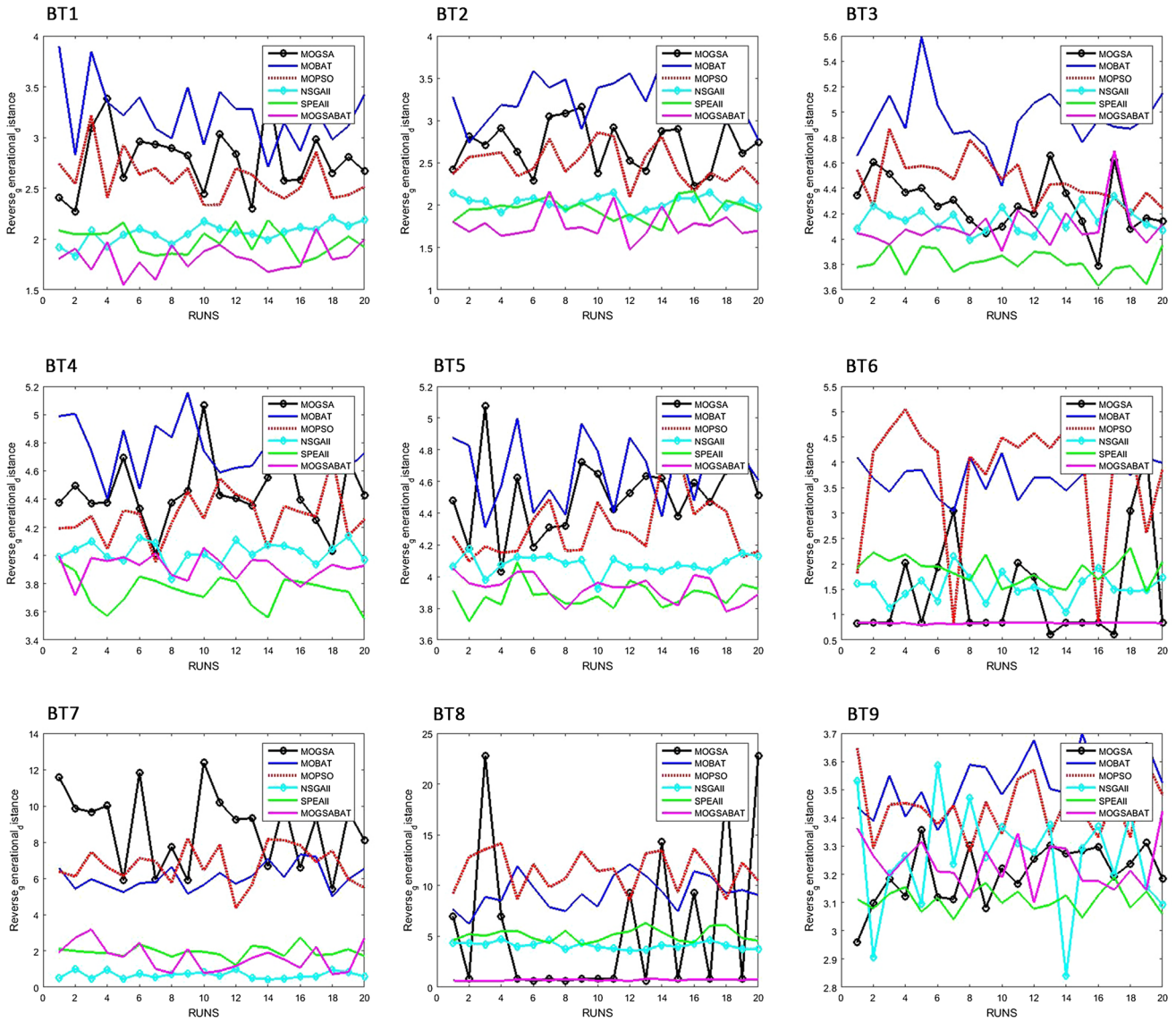

Fig. 6 Visualisation of RGD performance metric by MOGSABAT against the five competitive algorithms in BT

and *SD* values in *UF*2 and *UF*3 instances. *SPEAII* obtains the most remarkable mean in instance *UF*7 and *SD* values in instances *UF*2, *UF*4 and *UF*7. The hybrid *MOGSABAT* algorithm shows the most remarkable mean value in instances *UF*1, *UF*5 and *UF*6. The *MOPSO* algorithm also presents preferable value in the instance *UF*4 for the mean criterion.

As shown in Fig. 4, the proposed *MOGSABAT* algorithm has better *RGD* performance than *MOGSA* and *MOBAT* algorithms for functions *UF*1 to *UF*7 if we take $a = 0.001$ (where a represents the statistical level). Although the proposed algorithm performs equally with the *MOPSO* algorithm in function *UF*4, it is better in functions *UF*2, *UF*3, *UF*5, *UF*6 and *UF*7. Finally, the hybrid algorithm is better compared with the *SPEAII* algorithm in functions *UF*1, *UF*3, *UF*4, *UF*5 and *UF*6.

Table 5 compares the algorithms and instances depending on the *GD*, *mean* and *SD* criteria. Results show *NSGAII* in two instances of multiple functions *BT*1 and *BT*7, and it presents good *SD* results for instances *BT*1, *BT*2, *BT*4, *BT*5 and *BT*7. The instances of multiple functions *BT*3 and *BT*9 indicate that *SPEAII* yields good mean results in instances *BT*4 and *BT*5. The hybrid *MOGSABAT* algorithm displays the most remarkable value in instance *BT*8 and improved mean values in instance *BT*2.

As shown in Fig. 5, our proposed *MOGSABAT* algorithm has better *GD* performance in comparison with the *MOGSA* algorithm for functions *BT*1 and *BT*7 if we take $a = 0.01$ (where a represents the statistical level). It only fails in functions *BT*1 and *BT*9 compared with the *MOPSO* algorithm. However, the proposed *MOGSABAT* algorithm performs better as compared with *MOBAT* algorithm in functions *BT*1, *BT*3, *BT*4, *BT*5 and *BT*7. It performs equally with the *NSGAII* algorithm in instances *BT*1, *BT*5 and *BT*6. Finally, the hybrid algorithm is better compared with the *SPEAII* algorithm in functions *BT*1, *BT*3, *BT*5 and *BT*6.

Table 6 compares the algorithms and instances depending on the *RGD*, *mean* and *SD* criteria. Results show the *NSGAII* in instances of multiple functions *BT*1, *BT*5 and *BT*7 for the *SD* criterion, and it presents good mean results for instance *BT*7. The instances of multiple functions *BT*6 and *BT*9 indicate that *SPEAII* yields good *SD* criterion results, and better results are shown in instances *BT*1, *BT*3, *BT*4, *BT*5 and *BT*9 in terms of mean. The hybrid *MOGSABAT* algorithm displays the most remarkable value in instances *BT*2, *BT*6 and *BT*8 in the mean criterion and improved *SD* values in instance *BT*8.

In Fig. 6, the proposed *MOGSABAT* algorithm has better *RGD* performance compared with the *MOGSA*, *MOBAT* and *MOPSO* algorithms for functions *BT*1 to *BT*9 if we take $a = 0.001$ (where a represents the statistical level). In functions *BT*1, *BT*5 and *BT*9, it performs better compared with the *NSGAII* algorithm. Finally, the proposed *MOGSABAT* algorithm is better than the *SPEAII* algorithm in functions *BT*2, *BT*5, *BT*6, *BT*7 and *BT*8.

5 Conclusion

A new hybrid GSA and BAT algorithm called *MOGSABAT* is proposed to solve *MOPs*. The proposed algorithm is compared with multi-objective GSA, multi-objective BAT algorithm, *NSGAII*, *MOPSO* and *SPEAII* to verify the efficiency of the proposed algorithm for solving *MOPs*. The numerical experiment results showed that the proposed algorithm is a promising and efficient algorithm. In comparison with the aforementioned algorithms, *MOGSABAT* can obtain the global minimum or near global minimum of the *MOPs* faster.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Abbott R, Albach D, Ansell S (2013) Hybridization and speciation. J Evolut Biol (Wiley Online Library) 267(2):229–246
2. Beheshti Z, Shamsuddin SMH (2014) Centripetal accelerated particle swarm optimization. Inf Sci 256:54–79
3. Coello CAC, Pulido GT, Lechuga MS (2004) Handling multiple objectives with particle swarm optimization. IEEE Trans Evolut Comput 8(3):256–279
4. Deb K (2001) Multi-objective optimization using evolutionary algorithms, vol 16. Wiley, New York
5. Deb K (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evolut Comput 6(2):182–197
6. Derrac S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evolut Comput 1(1):3–18
7. Dorigo M, Maniezzo V, Colorni A (1996) The ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B 26(1):29–41
8. Hassanzadeh HR, Rouhani M (2010) A multi-objective gravitational search algorithm. In: Proceedings of the 2010 second international conference on computational intelligence, communication systems and networks (CICSyN). IEEE, pp 7–12. https://doi.org/10.1109/CICSyN.2010.32.
9. Huo, J. and Liu, L. (2018). Application research of multi-objective Artificial Bee Colony optimization algorithm for parameters calibration of hydrological model. Neural Comput Appl (2018). https://doi.org/10.1007/s00521-018-3483-4
10. Karimi-Nasab M, Ghomi SMTF (2012) Multi-objective production scheduling with controllable processing times and sequence-dependent setups for deteriorating items. Int J Prod Res 50(24):7378–7400

11. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, vol 4, pp 1942–1948
12. Kirkpatrick S, Gelatto CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:67–680
13. Li H, Zhang Q, Deng J (2017) Biased multiobjective optimization and decomposition algorithm. IEEE Trans Cybern 47(1):52–66
14. Liu H-L, Chen L, Deb K, Goodman ED (2017) Investigating the effect of imbalance between convergence and diversity in evolutionary multiobjective algorithms. IEEE Trans Evolut Comput 21(3):408–25
15. Ning J, Liu T, Zhang C (2018) A food source-updating information-guided artificial bee colony algorithm. Neural Comput Appl 30:775. https://doi.org/10.1007/s00521-016-2687-8
16. Nobahari H, Nikusokhan M, Siarry P (2012) A multi-objective gravitational search algorithm based on non-dominated sorting. Int J Swarm Intell Res 3:32–49
17. Nobahari H, Nikusokhan M, Siarry P (2012) A multi-objective gravitational search algorithm based on non-dominated sorting. Int J Swarm Intell Res 3(3):32–49. https://doi.org/10.4018/jsir.2012070103
18. Peng G (2016) Multi-objective particle optimization algorithm based on sharinglearning and dynamic crowding distance. Opt Int J Light Electron Opt 127(12):5013–5020
19. Prakash S, Trivedi V, Ramteke M (2016) An elitist non-dominated sorting bat algorithm NSBAT-II for multi-objective optimization of phthalic anhydride reactor. Int J Syst Assur Eng Manag 7(3):299–315
20. Ramli MR, Abas ZA, Desa MI, Abidin, ZZ, Alazzam MB (2018) Enhanced convergence of Bat Algorithm based on dimensional and inertia weight factor. J King Saud Univ Comput Inf Sci. https://doi.org/10.1016/j.jksuci.2018.03.010
21. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248
22. Rashedi E, Rashedi E, Nezamabadi-pour H (2018) A comprehensive survey on gravitational search algorithm. Swarm Evolut Comput 41:141–158
23. Reyes-Sierra M, Coello CC (2006) Multi-objective particle swarm optimizers: a survey of the state-of-the-art. Int J Comput Intell Res 2(3):287–308
24. Sabri NM, Puteh M, Mahmood MR (2013) A review of gravitational search algorithm. Int J Adv Soft Comput Appl 5(3):1–39
25. Silverman BW (1986) Density estimation for statistics and data analysis. CRC Press, London, p 26
26. Sun G, Zhang A, Jia X, Li X, Ji S, Wang Z (2016) DMMOGSA: diversity-enhanced and memory-based multi-objective gravitational search algorithm. Inf Sci 363:52–71
27. Tang KS, Man KF, Kwong S, He Q (1996) Genetic algorithms and their applications. IEEE Signal Process Mag 13(6):22–37
28. Tharakeshwar T, Seetharamu K, Prasad BD (2017) Multi-objective optimization using bat algorithm for shell and tube heat exchangers. Appl Therm Eng 110:1029–1038
29. Xiao J, Li W, Liu B (2018) A novel multi-population co-evolution strategy for single objective immune optimization algorithm. Neural Comput Appl 29:1115. https://doi.org/10.1007/s00521-016-2507-1
30. Yang X-S (2010) A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization. Studies in Computational Intelligence, vol 284. Springer, Berlin, pp 65–74
31. Yang X-S (2011) Bat algorithm for multi-objective optimisation. Int J Bio-Inspir Comput 3(5):267–274
32. 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, vol 264, pp 52–66
33. Zitzler E (1999) Evolutionary algorithms for multiobjective optimization: methods and applications. Ph.D. Thesis. Swiss Federal Institute of Technology, Zurich, Switzerland
34. Zitzler E, Laumanns M, Thiele L (2001) Improving the strength Pareto evolutionary algorithm. Eidgenssische Technische Hochschule Zrich 103