Knowledge and Information Systems
https://doi.org/10.1007/s10115-020-01503-x

REGULAR PAPER

Multi-objective Bonobo Optimizer (MOBO): an intelligent heuristic for multi-criteria optimization

Amit Kumar Das[1] · Ankit Kumar Nikum[2] · Siva Vignesh Krishnan[1] · Dilip Kumar Pratihar[1]

Received: 5 November 2019 / Revised: 29 July 2020 / Accepted: 31 July 2020

Abstract
Non-traditional optimization tools have proved their potential in solving various types of optimization problems. These problems deal with either single objective or multiple/many objectives. Bonobo Optimizer (BO) is an intelligent and adaptive metaheuristic optimization algorithm inspired from the social behavior and reproductive strategies of bonobos. There is no study in the literature to extend this BO to solve multi-objective optimization problems. This paper presents a Multi-objective Bonobo Optimizer (MOBO) to solve different optimization problems. Three different versions of MOBO are proposed in this paper, each using a different method, such as non-dominated sorting with adaptation of grid approach; a ranking scheme for sorting of population with crowding distance approach; decomposition technique, wherein the solutions are obtained by dividing a multi-objective problem into a number of single-objective problems. The performances of all three different versions of the proposed MOBO had been tested on a set of thirty diversified benchmark test functions, and the results were compared with that of four other well-known multi-objective optimization techniques available in the literature. The obtained results showed that the first two versions of the proposed algorithms either outperformed or performed competitively in terms of convergence and diversity compared to the others. However, the third version of the proposed techniques was found to have the poor performance.

Keywords Multi-objective optimization · Bonobo Optimizer · Intelligent algorithm · Multi-objective Bonobo Optimizer · Multi-criteria optimization

✉ Dilip Kumar Pratihar
dkpra@mech.iitkgp.ac.in

Amit Kumar Das
amit.besus@gmail.com

Ankit Kumar Nikum
nikumankit@gmail.com

Siva Vignesh Krishnan
sivavigneshk@gmail.com

[1] Department of Mechanical Engineering, Indian Institute of Technology Kharagpur, Kharagpur 721302, India

[2] Department of Mechanical Engineering, SVNIT, Surat 395007, India

Published online: 25 August 2020

1 Introduction

Multi-objective optimization problems (MOPs) are referred to those ones, which have more than one objectives. A multi-objective optimization problem characterized by $M(\geq 2)$ objectives, n variables, J-equality and K-inequality constraints can be formulated as follows (assuming minimization problem, without loss of generality):

$$\text{Minimize} \quad F(X) = (f_1(x), \ldots, f_M(x))^{\mathrm{T}} \tag{1}$$

subject to

$$\begin{aligned} & g_j(x) = 0, \quad j = 1, \ldots, J \\ & h_k(x) \leq 0, \quad k = 1, \ldots, K \\ & \text{where} \quad x = (x_1, \ldots, x_n)^{\mathrm{T}} \\ & \text{and} \quad x_i^L \leq x_i \leq x_i^U, \quad i = 1, \ldots, n \end{aligned}$$

A decision maker (DM) is a person, who gives preference information to obtain Pareto-front. MOPs can be classified based on the role of DM as follows:

- No preference method, where there is no role of DM,
- A priori method, where DM gives the preferences before optimization,
- A posteriori method, where DM chooses the best solution from a list of Pareto-optimal solutions,
- Interactive methods, where DM guides the search by giving the preferences iteratively during optimization.

The priori and posteriori methods are very popular among these methods [1]. A priori method optimizer combines the objectives of the MOP using weights provided by DM and uses a single-objective method to solve it. This yields a single optimum solution to the MOP. However, this method is found to have a major drawback. There is a chance that it may miss out the potential solutions, as there is no feedback to the preferences given by DM. Due to this reason, many researchers had turned their eyes from priori to posteriori methods for solving MOPs. Even though the computational cost is more, a posteriori allows DM to see which of the solutions are feasible and it guarantees that no superior solution is missed. Contrary to a priori method, optimization using a posteriori method leads to not one but more than one solutions. This obtained set of solutions is termed as Pareto-front (PF).

The set of all $x = (x_1, \ldots, x_n)^{\mathrm{T}}$, which satisfies all $(J + K)$ constraints, defines the feasible decision space (Ω). $F(x) \in R^M$ defines the attainable objective space. If the objectives in an MOP are conflicting in nature, then no single solution can optimize all the objectives simultaneously. Hence, the best trade-offs among the objectives can be obtained in terms of Pareto-optimality.

Definition 1 (*Pareto Dominance*) A solution vector x is said to dominate another solution vector y (denoted by $x \succ y$), if and only if,

$$(\forall i \in \{1, 2, \ldots, m\} : f_i(x) \leq f_i(y)) \wedge (\exists j \in \{1, 2, \ldots, m\} : f_j(x) < f_j(y)). \tag{2}$$

Definition 2 (*Pareto-optimal Solution*) A solution x is declared to lie in Pareto-optimal front, if and only if,

$$\neg \exists y \in \Omega : y \succ x. \tag{3}$$

Definition 3 (*Pareto-optimal Set*) The set PS that contains all Pareto-optimal solutions

$$\mathrm{PS} = \{x | \neg \exists y \in \Omega : y \succ x\}. \tag{4}$$

Definition 4 (*Pareto-optimal Front*) A front consisting of the values of all objective functions corresponding to Pareto-optimal solutions, as given below.

$$\mathrm{PF} = \{F(x) = (f_1(x), f_2(x), \ldots, f_m(x))^{\mathrm{T}} | x \in \mathrm{PS}\}. \tag{5}$$

Solving MOPs requires addressing of two conflicting goals: minimizing the distance between the obtained and true Pareto-front (i.e., convergence) and maximizing the spread of solution along the Pareto-front (i.e., diversity). A good optimizer takes these two things into consideration. Since the introduction of stochastic optimization techniques in 1984 [2], there has been significant development in multi-objective evolutionary/heuristic algorithms (MOEAs). Their popularity had increased, when compared to their classical counterparts, as they are able to find multiple Pareto-optimal solutions in one simulation run. Multi-objective optimization has various applications in science, engineering, economics, finance, operations research, chemistry, biology, construction and many other domains. A few examples of real-world multi-objective optimization problems are as follows: design and optimization of marine-protected area network [3], optimization of renewable-energy system [4], MOP for materials discovery [5], fuel consumption and NO_x emission optimization in automobile industry [6], and others.

The literature shows that several multi-objective evolutionary algorithms, such as Non-dominated Sorting Genetic Algorithm (NSGA-II) [7] and Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) [8], and multi-objective swarm intelligent algorithms, like Multi-Objective Particle Swarm Optimization (MOPSO) [9], Multi-Objective Grey-Wolf Optimizer (MOGWO) [10], Multi-Objective Ant Lion Optimizer (MOALO) [11], etc., were able to find true Pareto-optimal solutions for many real-world multi-objective optimization problems effectively. However, the base optimization algorithms for these MOPs, such as Genetic Algorithm [12] for NSGA-II and Particle Swarm Optimization [13] for MOPSO, are not found to be intelligent and adaptive enough. There are several metaheuristics proposed in recent years also. New techniques based on search and rescue operation [14], Emperor Penguins Colony algorithm [15], Black widow optimization algorithm [16], Mayfly optimization algorithm [17] and Red deer algorithm [18] are some of the examples of the very recently proposed nature-inspired optimization methods. For more information regarding other optimization techniques, interested readers can go through these research papers: [19, 20]. In general, the controlling parameters of these algorithms are observed to work with their fixed initial values. Therefore, these algorithms are not flexible enough to divert their focuses on either exploration or exploitation according to the need of the situation. Considering these issues, an intelligent and adaptive optimization algorithm, namely Bonobo Optimizer (BO) [21], has been proposed, recently. The controlling parameters of the proposed BO were made self-adjustable. Moreover, several new strategies, such as fission–fusion selection scheme, four different well-designed processes for generating offspring and consideration of two distinct phases made the search process of the developed BO more efficient and robust. It had been seen that the proposed BO was able to perform better compared to several other popular optimization algorithms, like GA, PSO and others in single-objective domain. However, there is no proposal for the multi-objective version of this efficient algorithm in the literature, till now. Besides this, a well-known theorem, namely "no free lunch (NFL)" [22], logically proves that there is no algorithm, which is able to solve all kinds of optimization problems with equal ease and efficiency. According to this

theorem, a superior performance of an algorithm on a set of problems does not guarantee the similar performance on a set of different problems. This theorem creates the direction for the researchers to propose new algorithms or to modify the existing algorithms for achieving the improved performances. Therefore, these are the motivations behind the novel work herein presented, in which we propose three different versions of Multi-objective Bonobo Optimizer (MOBO), named as MOBO-I, MOBO-II, and MOBO-III based on the recently published BO algorithm.

Thus, the main contributions of this paper can be summarized as follows:

(1) The recently proposed Bonobo Optimizer (BO), which had been developed for solving single-objective optimization problems, was converted into the multi-objective frameworks to solve MOPs.
(2) Multi-objective Bonobo Optimizer (MOBO) was developed using three different approaches. The first approach works based on the concept of non-dominated sorting and adopts the grid index approach for increasing diversity. The second approach uses the ranking scheme for sorting of the population and crowding distance approach to maintain diversity. The last approach follows the decomposition technique, in which Pareto-optimal solutions are obtained by dividing the multi-objective function into several single-objective functions.
(3) An extensive and detailed analysis on the performance of the proposed algorithms on a number of benchmark test problems was presented. In addition, the performances of all the three proposed algorithms were assessed with their popular competitors, such as NSGA-II, MOPSO, MOEA/D, and MOGWO.
(4) The performances of the proposed algorithms were evaluated based on the performance matrices, like maximum spacing (MS), spacing metric (SP), and inverse generational distance (IGD). Moreover, statistical analyses had been carried out with multiple comparisons of the obtained results.

The remainder of the paper is organized as follows: In Sect. 2, related studies on MOPs are summarized. In Sect. 3, a brief review of Bonobo Optimizer (BO) and then three versions of MOBO are proposed separately. Section 4 deals with the brief discussion on the performance indices used in this study. The benchmark test problems utilized for the performance evaluations of the algorithms are given in Sect. 5. A detailed quantitative and qualitative performance analysis along with the relevant discussion is presented in Sect. 6. Finally, some conclusions are drawn in Sect. 7.

2 Literature review

The major goal of multi-objective optimization algorithm is to find diverse solutions lying on the Pareto-front. This allows the decision maker (DM) to choose the preferable one from a wide range of feasible and optimal solutions. In the past, some methods were used, where a multi-objective problem was transformed into a single-objective one and then, it was solved. The global criterion method uses minimization of distance between multiple reference points and viable destination areas to convert multi-objectives problem into a single-objective one [23]. In weighted sum approach, a multi-objective problem is transformed into single-objective problem using weights given by DM and then, a single-objective optimizer is used to solve it [24]. Though it is simple and easy to use, it has two problems: difficulty in choosing the appropriate weights and inability to determine the non-convex regions of Pareto-front. To overcome these difficulties in solving non-convex problems, $\varepsilon\varepsilon$-constraint method is used

[1]. It optimizes only one objective at a time, while the others are converted into the limits. However, it has its own limitations. There are many studies in the literature that had tried to overcome the problems posted by a priori method but could not overcome all the problems.

Classical multi-objective methods try to transform multi-objective problem into a single-objective problem. However, in most of the real-world multi-objective problems, suitable solutions are rarely found by these classical methods, since the objectives usually clash with each other. Multi-objective Evolutionary Algorithms (MOEAs) overcome some of the problems, such as infeasible areas, local fronts, isolation of optimum solution, and diversity of solution. These MOEAs, due to their population-based nature, are able to approximate the whole Pareto-front (PF) of an MOP in a single run. Moreover, since the information regarding the search space is transferred among the search agents, they are able to converge to true Pareto-front faster. For the past three decades, many MOEAs and nature-inspired metaheuristic algorithms had been developed to solve real-world multi-objective optimization problems. Some of the well-performing algorithms are described below, in brief.

Strength Pareto Evolutionary Algorithm (SPEA) proposed by Zitzler and Thiele [25] stores the Pareto-optimal solutions found so far in an external archive. The algorithm uses Pareto dominance to assign fitness values to the individuals, and these fitness values are determined only from the non-dominated solutions in the external archive. Then, it performs clustering to reduce number of non-dominated solutions stored in external archive without destroying the characteristics of the Pareto-front. In SPEA, all solutions in the archive participate in selection and a new niching method is used that does not require any fitness sharing parameter. In SPEA, the Pareto-optimal front may not able to converge due to high selection pressure of the elites, if a large external population is considered. On the contrary, effect of elitism will be not be proper, if a small external population is assigned [26].

Pareto Archived Evolutionary Strategy (PAES) developed by Knowles and Corne [27] introduced a new adaptive grid algorithm with reduced computational cost. The algorithm does not require any critical setting of niche-size parameter. It also uses external archive to store non-dominated solutions found so far. If the archive is full, the adaptive grid mechanism is triggered. The objective space is divided into a number of grids. If the obtained solution is observed to be outside of the grids, the grids are recalculated to cover it. However, if the obtained solution lies within the grid, it is said to be the part of grid, which has the lowest number of particles. This grid mechanism helps in maintaining diversity in the archive. One of the disadvantages of PAES is that with the increase of depth parameter, the number of hypercube also increases exponentially and thus leads to an uncontrolled spread of Pareto-optimal solutions [26].

Non-dominated Sorting Genetic Algorithm-II (NSGA-II), proposed by Deb et al. [7], is a multi-objective version of the GA and a successor of NSGA [28]. It had addressed the problem posed by NSGA by replacing the user-defined sigma-share (σ) for diversity preservation with the density estimation of each individual in the population and estimation of crowding distance. Besides this, it reduced the computational complexity from $O\left(MN^3\right)$ to $O\left(MN^2\right)$, after adding elitism by forming a new temporary population of size $2N$ that includes the parent and offspring population. Here, M and N denote the number of objectives and population size, respectively. NSGA-II starts with an initialization of random population followed by non-dominated sorting. Then, individuals are selected depending on their ranks and crowding distances. Stochastic operators, like crossover, mutation are then applied to the selected population. Next, a recombination of the parent and offspring population is tried. Then, from the combined population of size $2N$, a population of size N is selected for the next generation based on the assigned ranks and crowding distances. This completes one

generation. It is to be mentioned here that NSGA-II is one of the most popular MOEAs reported in the literature.

Another popular algorithm for solving MOP is Multi-objective Particle Swarm Optimization (MOPSO) [9]. It is the multi-objective version of PSO [13], which works based on the swarm intelligence of the particles. MOPSO is also a population-based approach, which uses the global repository and implements a geographically based method to maintain diversity. The historical record of the best solution found by a particle could be used to store non-dominated solutions generated in the past. In this way, elitism is kept in MOPSO. In addition, mutation is also applied, so that premature convergence with the false Pareto-front does not occur.

Multi-objective Evolutionary Algorithm based on Decomposition (MOEA/D) [8] is also another popular algorithm for solving MOPs. In MOEA/D, an MOP is transformed into a set of scalar optimization subproblems through the application of decomposition approaches. Then, an evolutionary algorithm is utilized to optimize these subproblems simultaneously. MOEA/D solves each subproblem by evolving a population of solutions. At each generation, the population contains the best solution found since the starting of the algorithm for each subproblem. The neighborhood relations among these subproblems are defined based on the distances between their aggregation coefficient vectors (i.e., the weights for the subproblems). The optimal solutions of two neighboring subproblems should be similar. Each subproblem is optimized by using information from its several neighboring subproblems. The problems, such as fitness assignment and diversity maintenance, are handled by the MOEA/D in a better way compared to other MOEAs without involving decomposition. MOEA/D has the less computational complexity than that of NSGA-II.

Besides these, several other multi-objective evolutionary algorithms and multi-objective swarm intelligent algorithms had been proposed. A Multi-objective Ant Colony Optimization algorithm developed based on elitist selection strategy [29], Multi-objective Grey-Wolf Optimizer (MOGWO) [10], Multi-objective Ant Lion Optimizer (MOALO) [11], Strength Pareto Evolutionary Algorithm Based on Reference Direction (SPEA/R) [30], Multi-objective Cooperative Salvo attack against group target [31], Lebesgue Indicator-Based Evolutionary Algorithm (LIBEA) [32], Multi-objective optimization algorithm based on lightning attachment procedure optimization algorithm (MOLAPO) [33], Evolutionary Multi-Objective Membrane Algorithm [34], Multi-objective Jaya Algorithm [35], Micro-Multi-objective Genetic Algorithm [36], Multi-objective Bird Swarm Algorithm [37], etc., are some of the other examples in this category. Interested readers may follow some of these review papers [38–41] on multi-objective optimization techniques for getting more information. Now, according to NFL theorem, it may be possible to develop a new algorithm that will be able to solve an unsolved problem reported in the literature or will be able to solve an already solved problem with an improved performance. Moreover, the recently developed Bonobo Optimizer (BO) is claimed to be an intelligent and superior-performing optimization technique with the self-adjusting controlling parameters. It has been observed that BO is performing well for a variety of test functions and real-world optimization problems. It is capable of maintaining a superior balance between population diversity and convergence. Therefore, it is quite possible that the multi-objective version of the proposed BO will have the better performance compared to other existing algorithms. Here, a multi-objective version of Bonobo Optimizer is proposed with three different approaches.

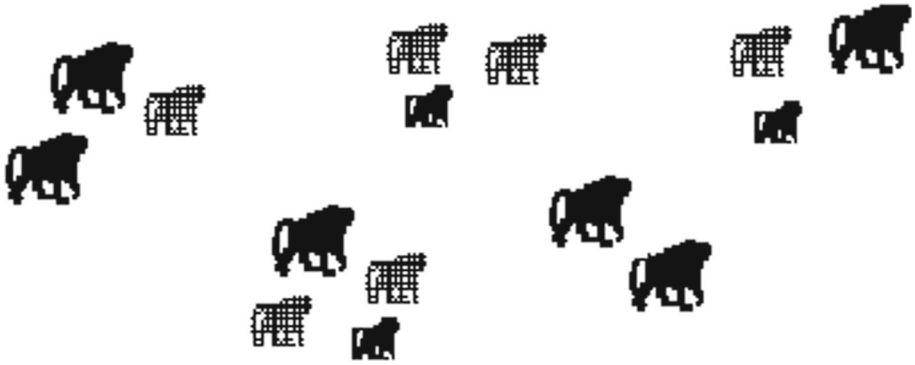

Fig. 1 Fission–fusion social groups of bonobos [42]

3 Multi-objective Bonobo Optimizer (MOBO)

3.1 A brief review of Bonobo Optimizer (BO)

Recently, an intelligent and efficient metaheuristic technique, namely BO, has been proposed by Das and Pratihar [21]. The social behavior and reproductive strategies of bonobos were the main inspiration for the development of this algorithm. The source code of BO is given in https://sites.google.com/site/softcomputinglaboratory/Home. bonobos follow fission–fusion social strategy, which is nothing but to divide into a few smaller groups (fission) and do several activities (refer to Fig. 1). Again, they reunite (fusion) to serve some specific activities, like sleeping together, etc. This unique strategy was copied into the algorithm to make the searching mechanism more efficient. Similar to other heuristics, BO is also a population-based algorithm. Each solution in the population is called a bonobo and the bonobo with the best rank in dominance hierarchy in the population is called the alpha-bonobo (α^{bonobo}). When there is an improvement in alpha-bonobo in the iteration, the phase is termed as positive phase (pp). On the contrary, when there is no improvement in alpha-bonobo, the phase is termed as negative phase (np). The counts of consecutive number of iterations of positive phase and negative phase are called positive phase count (ppc) and negative phase count (npc), respectively. N is the population size.

A step-by-step working principle of BO algorithm is described below.

Step 1 Initialization of non-user-defined parameters

Initially, some of the parameters are set as follows: ppc $= 0$, npc $= 0$, cp $= 0$, $p_{\text{xgm}} = p_{\text{xgm\_initial}}$, $p_{\text{p}} = 0.5$, $p_{\text{d}} = 0.5$, $\text{tsgs}_{\text{factor}} = \text{tsgs}_{\text{factor\_initial}}$, where ppc, npc, cp, p_{xgm}, $\text{tsgs}_{\text{factor}}$, p_{p}, and p_{d} denote the positive phase count, negative phase count, change in phase, probability of extra-group mating, temporary subgroup size factor, probability of phase, and directional probability, respectively.

Step 2 Selection of bonobo using fission–fusion technique

For selecting the pth-bonobo, which is going to participate in mating with the ith-bonobo, the maximum size of the subgroup (tsgs_{max}) is calculated at first using Eq. (6).

$$\text{tsgs}_{\text{max}} = \text{maximum}\left(2, \text{tsgs}_{\text{factor}} \times N\right) \tag{6}$$

Now, ith-bonobo participates in mating with pth-bonobo to create the offspring. To select pth-bonobo, a temporary subgroup of size tsgs (any value between 2 and tsgs_{max}) is formed by randomly choosing non-duplicate bonobos from the population of size of $(N-1)$ (where

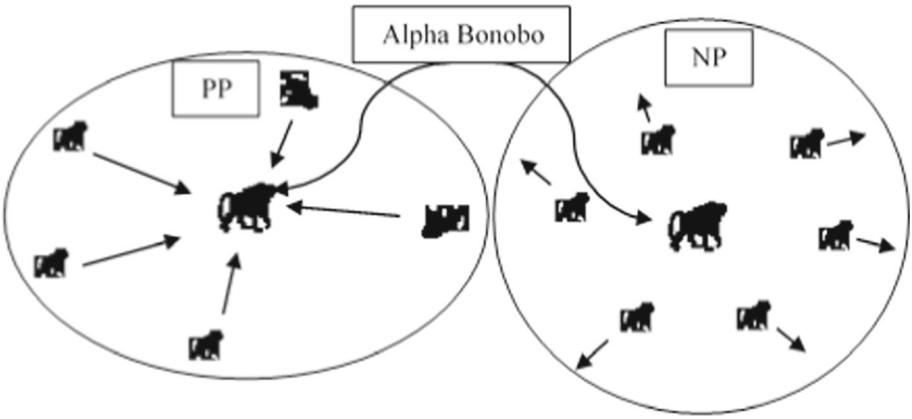

Fig. 2 Directions of movements of different bonobos in PP and NP with the higher probabilities

*i*th-bonobo is not present). The best solution from the subgroup is chosen as the *p*th-bonobo, if its fitness value is found to be better than that of the *i*th-bonobo; otherwise, a random bonobo from the subgroup is chosen as *p*th-bonobo.

Step 3 Creation of bonobo using different mating strategies

The mating strategy varies according to the condition of phase. If it is a positive phase, the chances of promiscuous and restricted matings are high. On the other hand, the possibilities of occurring consortship and extra-group matings are found to be high in a negative phase.

If a random number (r), generated in the range of (0, 1), is seen to be either less than or equal to the phase probability (p_{p}), a new bonobo is created following Eq. (7) (promiscuous and restricted mating)

$$\begin{aligned}\mathrm{new\_bonobo}_j = \mathrm{bonobo}_j^i + r_1 \times \mathrm{scab} \times \left(\alpha_j^{\mathrm{bonobo}} - \mathrm{bonobo}_j^i\right) + (1 - r_1) \\ \times \mathrm{scsb} \times \mathrm{flag} \times \left(\mathrm{bonobo}_j^p - \mathrm{bonobo}_j^i\right),\end{aligned} \tag{7}$$

where j varies from 1 to d. Here, d is the number of decision variables of the optimization problem to be solved. Sharing coefficients for the alpha-bonobo and *p*th-bonobo are represented using scab and scsb, respectively. The parameter, "flag" can take only two distinct values: 1 (for promiscuous type of mating) or − 1 (for restrictive type of mating). r_1 is also a random number lying in the range of (0, 1). In another case, where the random number r is observed to be greater than p_{p}, either extra-group or consortship mating strategy is applied to yield an offspring. In a positive phase (PP), the rest of the bonobos in the population follow the alpha-bonobo, whereas the bonobos try to move away from the alpha-bonobo in case of a negative phase (NP) (refer to Fig. 2).

Step 4 Variable boundary limiting conditions

If the generated offspring is found to cross the variable's boundary limits, then its value is restricted to the nearest boundary limit.

Step 5 Acceptance of new bonobo and updating of alpha-bonobo

The created offspring is accepted, when it is found be better than the parent one in terms of fitness value or a random number lying in the range of (0, 1) is observed to be either less than or equal to the probability of extra-group mating (p_{xgm}). Moreover, if the yielded offspring is seen to be better compared to the alpha-bonobo, then the created new bonobo is regarded as the alpha-bonobo.

Step 6 Updation of controlling parameters

If the newly obtained alpha-bonobo is found to be better than that of the previous iteration, the parameters are updated as follows:.

$$\begin{aligned}
&\text{npc} = 0, \text{ppc} = \text{ppc} + 1, \text{cp} = \text{minimum}(0.5, \text{ppc} \times \text{rcpp}),\\
&p_{\text{xgm}} = p_{\text{xgm_initial}}, p_{\text{p}} = 0.5 + \text{cp}, p_{\text{d}} = p_{\text{p}},\\
&\text{tsgs}_{\text{factor}} = \text{minimum}\left(\text{tsgs}_{\text{factor_max}}, \left(\text{tsgs}_{\text{factor_initial}} + \text{ppc} \times \text{rcpp}^2\right)\right),\\
&\text{tsgs}_{\text{factor_initial}} = 0.5 \times \text{tsgs}_{\text{factor_max}}.
\end{aligned}$$

Otherwise, the parameters are updated in the following manner:

$$\begin{aligned}
&\text{ppc} = 0, \text{npc} = \text{npc} + 1, \text{cp} = -\text{minimum}(0.5, \text{npc} \times \text{rcpp})\\
&p_{\text{xgm}} = \text{minimum}\left(0.5, \left(p_{\text{xgm_initial}} + \text{npc} \times \text{rcpp}^2\right)\right), p_{\text{p}} = 0.5 + \text{cp},\\
&p_{\text{d}} = p_{\text{p}}, \text{tsgs}_{\text{factor}} = \text{maximum}\left(\text{tsgs}_{\text{factor_max}}, \left(\text{tsgs}_{\text{factor_initial}} - \text{npc} \times \text{rcpp}^2\right)\right).
\end{aligned}$$

3.2 Proposed algorithms

As discussed earlier, the uniqueness of the proposed Multi-objective Bonobo algorithm (MOBO) lies in the essence of the single-objective BO algorithm, which was developed based on the living style of bonobos. The proposed MOBO algorithm developed based on its original form creates a random population of size N and then improves this population over a number of iterations replicating the bonobo's social lifestyle. In the single-objective version, this algorithm finds the globally optimum solution. However, it tries to search for a set of non-dominated solutions in the multi-objective version. Therefore, the basic idea of non-dominated sorting [28] has been used here. It sorts the solutions into the non-dominated and dominated sets. In this study, three versions of MOBO have been proposed.

3.2.1 MOBO-I

The first version of the proposed Multi-objective Bonobo Optimizer (MOBO) is termed as MOBO-I. This version is inspired from the Multi-objective Particle Swarm Optimization (MOPSO) [9] algorithm. It uses the non-dominated sorting method to create non-dominated and dominated solutions. One repository is generated, which contains the non-dominated solutions. This repository is used to save the non-dominated solutions created so far, and it deletes the dominated ones in order to update the repository. Its size is kept fixed and is generally taken half of the size of current population. This repository collects the solutions from the existing population itself and the updating of the same is done by comparing the newly generated solution with the existing solutions in the population [10]. Moreover, for updating of the repository, a few conditions are applied and these are as follows:

- If the new solution is dominated by at least one of the archive residences, it should not be allowed to enter the repository.
- When a new solution dominates one or more solutions in the repository, the dominated solution(s) in the archive should be removed and the new one is allowed to enter the repository.
- If neither the new solution nor repository member dominates each other, the new solution is added to the repository.

In this algorithm, the concept of grid mechanism is used to maintain the diversity and uniformity in the population. By this function, the whole search space is divided into $K + 2$ number of grids, where K is the number of grids specified by the user. The more the number of grids, the better it can maintain the diversity. The grids contain the region from negative infinity to positive infinity, and the grids are equally spaced. Now, each grid is identified by a grid index number. Each element of the population is allotted a grid index number by an algorithm to identify the grid in which the solution lies.

In the single-objective BO, the alpha-bonobo is selected based on the fitness of the solution. However, no such concept exists in the multi-objective version. Therefore, to select the leader or the alpha-bonobo, a Roulette wheel selection method [43] is used on the non-dominated solutions of the repository. The selection probability ($p_{\text{selection}}$) is measured using Eq. (8), as follows:

$$p_{\text{selection}} = C/n, \tag{8}$$

where C is a constant and n is the number of solutions present in different grids [9]. Hence, the probability of selection of the alpha-bonobo from the grid with more solutions is less. In this manner, this mechanism helps in maintaining the diversity and uniformity of the Pareto-front.

In BO, the rest of the bonobos are observed to follow the alpha-bonobo in case of promiscuous or restrictive kind of mating. Moreover, the selection of alpha-bonobo is carried out once in an iteration. However, in case of MOBO-I, this task is performed for each of the solutions (i.e., N times, where N is the population size) in an iteration. Therefore, it is going to increase the diversity in the offspring population.

Besides this, the grid mechanism also helps in deleting extra solution from the repository. The probability of a solution getting selected for deletion (p_{deletion}) is calculated as follows:

$$p_{\text{deletion}} = n/C, \tag{9}$$

where n is the number of solutions in a particular grid and C is a constant. It is obvious that for the deletion of the extra members, the probability of selection of the grid with more number of solutions is high. This helps in maintaining the diversity of the population and increases the coverage of the solution. Therefore, by grid mechanism technique, the solutions from the more crowded areas are selected for deletion.

The selection of pth-bonobo is carried out based on repository solutions. If the size of the temporary subgroup (tsgs) is found to be greater than or equal to that of the repository, a randomly chosen solution from the repository is regarded as the pth-bonobo. On the other hand, if tsgs is observed to be less than the repository size, the members of the temporary subgroup are chosen randomly from the repository. Then, the pth-bonobo is selected from that temporary subgroup at random.

Acceptance criterion for the offspring

Here, the offspring is accepted, if it is created using either promiscuous or restricted mating strategy. In case, the offspring is generated using either consortship or extra-group mating strategy; it is accepted, only if it satisfies any one of the following two conditions:

Condition 1 The offspring dominates the parent solution.

Condition 2 $r_2 \leq p_{xgm}$, where r_2 is a random number generated in the range of (0, 1) and p_{xgm} is the probability of extra-group mating.

Here, another concept, namely diversity quotient (dq), is added in the algorithm. The term dq is measured as follows: At first, the ratio of the standard deviation of each objective

function to the difference of the maximum and minimum values of the same objective is calculated. This calculation is done only on the repository members. Now, the average of these ratios is termed as dq, which is given below.

$$\mathrm{dq} = \frac{\sum_{i=1}^{m} \frac{\mathrm{SD}_i}{\left(\mathrm{Obj}_i^{\max} - \mathrm{Obj}_i^{\min}\right)}}{m} \tag{10}$$

Here, SD_i is the standard deviation of the ith-objective function present in the repository and i varies from 1 to m. $\mathrm{Obj}_i^{\max}$ and $\mathrm{Obj}_i^{\min}$ are the maximum and minimum values of the ith-objective function present in the repository, respectively. Now, a phase is termed as the positive phase, if the two conditions mentioned below are satisfied simultaneously.

Condition 1 The number of solutions present in the repository at the current iteration is found to be either greater than or equal to that of the previous iteration.

Condition 2 The calculated value of dq at the current iteration is observed to be greater than that of the previous one.

If these stated conditions are not fulfilled at a time, the phase is declared as a negative phase. However, if the number of element in the repository is found to be equal to 1, the phase is directly considered as the negative one. It is to be noted here that these two conditions are designed in such a way that it will help to increase the number of elements in the repository and enhance the diversity in the population of the repository. After determining the type of phase, the parameters of MOBO-I are updated similar to that of the BO. However, the values of the directional probability (p_d) and "flag" are taken as constants and these are found to be equal to 0.5 and 1, respectively. This is done to incorporate more diversity in the population. Moreover, this described process of determining the type of phase is applied across all the proposed MOBOs. The pseudo-code for MOBO-I is given in Fig. 3.

The parameters like scab, scsb, $\mathrm{tsgs}_{\mathrm{factor\_max}}$, $p_{\mathrm{xgm\_initial}}$ are the key factors behind maintaining both the diversity and convergence of the Pareto-front. scab and scsb are the parameters used in either promiscuous or restrictive mating strategy [21]. With the higher value of these parameters, the diversity in population is expected to increase, whereas the convergence capability of the search mechanism decreases. In a similar way, with the increase of $p_{\mathrm{xgm\_initial}}$, the diversity in the population is found to increase and vice versa. However, exploitation power is observed to be high with the larger value of $\mathrm{tsgs}_{\mathrm{factor\_max}}$ and vice versa. Hence, all these parameters should be set in such way that there should a proper balance between diversity and convergence. Therefore, a parametric study is performed to set these parameters, which is discussed in Sect. 6.

3.2.2 MOBO-II

The second version of this multi-objective algorithm is inspired from NSGA-II [7] algorithm. It inherits the concept of non-dominated sorting, assigning ranks to population and crowding distance from NSGA-II algorithm. The main difference between this version and the version developed using grid mechanism (that is, MOBO-I) lies with the sorting of population based on the ranks and crowding distances instead of grid to maintain the diversity. Moreover, the solutions from the least crowded space are given the preferences.

Sorting of the population is done using a ranking scheme, where the non-dominated solutions are given rank 1, at first. Next, again the non-dominated solutions are found from the existing dominated solutions and these are assigned rank 2. In this way, all the solutions

Initialize the bonobo population X_i *(i = 1, 2, ..., N)*
Initialize the parameters of BO
Calculate the objective values for each bonobo
Create the grid for search space
Find the non-dominated solutions and initialized the archive with them
iteration=1
While *(iteration < Max number of iterations)*
  For *each bonobo*
   Select α^{bonobo}
   Select p^{th}*-bonobo*
   Find the new bonobo using suitable mating strategy
   Calculate the objective function values
   Accept or reject the new bonobo
  End
   Find the non-dominated solutions in the current population
   Add non-dominated solutions to the repository
   Keep only the unique points in the repository
   Find the non-dominated solutions in the repository
   If *the repository is full*
   Run the grid mechanism to omit extra members of the repository
  End
  Update the grid of the repository members
  Calculate dq *of current repository and number of solutions present in repository*
   Determine the type of phase
   Update the parameters of BO according to type of phase
End

Fig. 3 Pseudo-code of MOBO-I

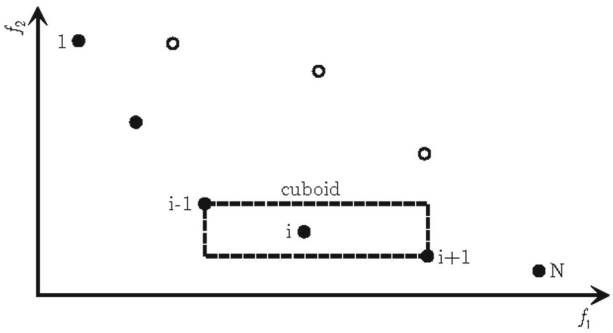

Fig. 4 A schematic representation of the method of crowding distance calculation

are provided certain ranks and this is known as the ranking scheme of NSGA-II. Crowding distance is the perimeter of the rectangle formed by the neighboring solutions of the *i*th-solution (i.e., $(i-1)$th and $(i+1)$th-solutions). Hence, the higher value of crowding distance indicates comparatively the less crowded space and vice versa. A schematic representation of the method of crowding distance calculation is shown in Fig. 4.

Thus, by using these two strategies, Pareto-front of non-dominated solution is found in NSGA-II algorithm and the same procedure is followed in this version of MOBO, namely MOBO-II.

Most of the strategies of MOBO-I and MOBO-II are found to be the same, except a few things. The differences between the above two algorithms are stated below.

- In MOBO-II, the leader is selected from the repository of non-dominated solutions on the basis of crowding distance. Hence, the probability of selecting a solution (P) is given below.

$$P \propto \text{CD}, \tag{11}$$

where CD is the crowding distance of the solution and the selection is done using Roulette-wheel selection method. Moreover, the crowding distances of the extremes are kept equal to the highest crowding distance of the rest of the solutions in the population. This is done to increase the probability of selection of other solutions too.

- Selection of pth-bonobo is carried out from the current repository at random in case of MOBO-I. However, in MOBO-II, it is done on the basis of ranks and CD.
- In MOBO-II, the created offspring of size N is combined with the parent solutions to obtain a mixed population of size $2N$. Next, the ranks and crowding distances (CD) of the mixed population are determined. From this $2N$ population, a population of size N is selected based on the ranks and CD of the solutions.
- More preference is given to the better rank in filling the new population. However, if the ranks are found to be the same, then the solution with the more CD is preferred. This strategy is adopted from NSGA-II algorithm.

A pseudo-code of MOBO-II is depicted in Fig. 5.

3.2.3 MOBO-III

The last approach used to create the multi-objective version of BO is the decomposition technique. This version of BO is inspired by the Multi-objective Evolutionary Algorithm based on Decomposition (MOEA/D). The basic principle of MOEA/D is to divide the multi-objective problem into a number of single-objective problems. The most traditional way of decomposition is nothing but applying the weights to the different objectives. However, there are many disadvantages of this traditional approach. If the Pareto-front (PF) is found to be concave in nature (convex in the case of minimization), this approach may work well. However, not each Pareto-optimal vector can be obtained by this approach, in the case of non-concave PFs. Therefore, other advanced techniques had been developed for decomposing the problem, namely Tchebycheff approach and Boundary Intersection (BI) approach [8]. Out of these two, Tchebycheff approach [44] is used to develop this algorithm. A brief description of Tchebycheff approach is given below.

Let $\lambda^1, \ldots, \lambda^Q$ be a set of even spread weight vectors and z^* be the reference point. As the main principle of this algorithm is to divide the PF into Q scalar optimization subproblems using the Tchebycheff approach, the objective function of the jth subproblem is expressed as follows:

$$g^{\text{te}}\left(x|\lambda^j, z^*\right) = \max_{1 \le i \le m} \left[\lambda_i^j \left| f_i(x) - z_i^* \right|\right], \tag{12}$$

Initialize the bonobo population X_i (i = 1, 2, ..., N)
Initialize the controlling parameters of BO
Calculate the objective values for each bonobo
Assign crowding distance and ranks to population
Select the repository with non-dominated solutions
iteration=1
While *(iteration < Max number of iterations)*
  For *each bonobo*
    Select alpha bonobo
    Select p^{th}-bonobo
    Find the new bonobo as done in single objective BO
  End
 Calculate the objective values of all new bonobos
 Find the mixed population of size 2N combining new bonobos and parent population
 Assign crowding distance and ranks to mixed population
 Find the current population of size N from the mixed population based on ranks and CD
 Select the repository with non-dominated unique solutions
 Calculate dq of current repository and number of solutions present in repository
 Determine the type of phase and update the controlling parameters
End

Fig. 5 Pseudo-code of MOBO-II

where $\lambda^j = \left(\lambda_1^j, \ldots, \lambda_m^j\right)^{\mathrm{T}}$. MOBO-III minimizes all these Q objective functions simultaneously in a single run. Note that the optimal solution of $g^{\mathrm{te}}(x|\lambda^i, z^*)$ should be close to that of $g^{\mathrm{te}}(x|\lambda^j, z^*)$, if λ^i and λ^j are close to each other. Hence, the neighboring solution close to λ^i is helpful in finding the optimal solution of the given problem. Therefore, the concept of neighboring solution is applied in MOBO-III, which is a motivation from MOEA/D itself.

In MOBO-III, a neighborhood of weight vector λ^i is defined as a set of the closest weight vectors in $\lambda^1, \ldots, \lambda^Q$. The neighborhood of the *i*th subproblem consists of all the subproblems with the weight vectors from the neighborhood of λ^i. The population is composed of the best solution found so far for each subproblem. Only the current solutions to its neighboring subproblems are exploited for optimizing a subproblem in MOBO-III. Apart from this, MOBO-III also maintains the decomposed cost "*g*" of each solution in population, which is the optimal solution of objective function of *j*th subproblem determined by the following equation:

$$g = \max_{1 \le i \le m} \left[\lambda_i^j \left| f_i(x) - z_i^* \right|\right], \tag{13}$$

After creating the subproblems, alpha-bonobo is selected based on the decomposed cost. For a minimization problem to be solved, the solution with the minimum decomposed cost is regarded as the alpha-bonobo. According to the algorithm, the alpha-bonobo has some user-defined number of the neighbors. In this set of neighbors, the alpha-bonobo solution is also included. During modification of each solution, a randomly chosen member of those neighbors is used as the alpha-bonobo in the equation. Besides this, the *p*th-bonobo is selected as the member of the temporary subgroup with the best decomposed cost. Next, *i*th-bonobo is improved in the way similar to BO and its corresponding child solution is obtained. Based on the acceptance criterion that the decomposed cost of the new solution should be less than the

decomposed cost of *i*th-bonobo, or a random number, generated in the range of (0, 1), should be smaller than p_{xgm}, the child solution replaces the *i*th-bonobo. Similarly, these criteria are also used to change the neighboring solutions of the *i*th-bonobo. Hence, if the child solution is found to be the better than *i*th-bonobo or its neighboring solutions, then it replaces them and a new population is formed.

At each generation, MOBO-III with the Tchebycheff approach maintains:

- A population of points, where $x^1, \ldots, x^n$ is the current solution to the *i*th subproblem.
- $\text{FV}^1, \ldots, \text{FV}^n$, where FV^i is the fitness value of x^i, i.e., $\text{FV}^i = \text{FV}(x^i)$, for each $i = 1, \ldots, n$.
- Decomposed cost of each solution g.
- $z = (z_1, \ldots, z_m)^{\text{T}}$, where z_i is the best value found so far for the objective f_i.
- An external population (EP), which is used to store non-dominated solutions found during the search.

The main differences between MOBO-III and MOBO-I are as follows:

- In MOBO-I, the comparison between two solutions is done on the basis of domination criteria. However, in MOBO-III, the comparison is done on the basis of the decomposed cost allotted to each solution, and the solution with less value of decomposed cost is preferred for a minimization problem.
- In MOBO-I, the new bonobo is created and accepted after its comparison with *i*th-bonobo, but in decomposition technique, all the neighboring solutions are replaced by new bonobo, if their decomposed costs are found to be better than that of new bonobo.

Hence, MOBO-III inherits all the characteristics of BO [21] algorithm. It also inherits all the parameters used in BO. Also, in all the multi-objective BO algorithms, we keep only unique points in the repository and discard all the stacked or repeated points. Therefore, this feature helps in maintaining the diversity and increasing the range of solutions. A pseudo-code for the proposed MOBO-III is given in Fig. 6.

Computational complexity The computational complexity of both MOBO-I and MOBO-II is of the order of $O(MN^2)$, where N is the number of individuals in the population and M is the number of objectives. The complexity was seen to be equal to that of other well-known algorithms, such as NSGA-II, MOPSO, SPEA2 [25], and PAES [27]. However, the computational complexity was found to be less than that of some of the algorithms, like NSGA [28] and SPEA [45]. For the case of MOBO-III, which uses the multi-objective framework of MOEA/D, it was found to have computational complexity similar to that of MOEA/D and that was equal to $O(MNT)$, where T is the number of neighboring solutions.

4 Performance indices

Two significant things to be considered in multi-objective optimization are: (i) convergence of Pareto-optimal set and (ii) maintaining diversity among the solutions of Pareto-optimal set. In this study, three different performance indices are used to measure these two things, which are described below.

(a) Inverse Generational Distance (IGD) [10]

IGD is employed to measure the convergence of the obtained Pareto-front. To calculate IGD, true Pareto-front must be available. It can be formulated as follows:

$$\text{IGD} = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}, \tag{14}$$

Initialize the bonobo population $X_i (i = 1, 2, ..., n)$
Initialize the controlling parameters of BO
Create sub problems and neighbours for each solution
Calculate the objective values for each search agent and value of z
Find the non-dominated solutions from the population and initiate repository with them
t=1
While *(t < Max number of iterations)*
  For *each search agent*
  Select alpha-bonobo
  Select p^{th}*-bonobo*
  Find the new bonobo by equations as done in single-objective BO
  Calculate the objective function values
  Accept or reject the new bonobo
 End
  Find the non-dominated solutions in the current population
  Add non-dominated solutions to the repository
  Keep only the unique points in the repository
  Find the non-dominated solutions in the repository
  If *the repository is full*
  Delete extra members of the repository randomly
  End
  Calculate dq *of current repository and number of solutions present in repository*
  Determine the type of phase
  Update the parameters of BO according to type of phase
End

Fig. 6 Pseudo-code of MOBO-III

where n represents the number of true Pareto-optimal solutions and d_i^2 indicates the Euclidean distance between the ith-true Pareto-optimal solution and the closest obtained Pareto-optimal solution in the reference set. In IGD, the Euclidean distance is determined for each true solution with respect to its nearest obtained Pareto-optimal solution in the objective space. The less the value of IGD, the better the Pareto-optimal set of the solutions obtained for a minimization problem.

(b) Maximum Spread (MS)

MS measures the distance between the boundary solutions in the obtained Pareto-optimal solutions, as proposed by Zitzler [25]. A normalized version of MS as suggested by Deb [46] has been employed in this paper for performance analysis. It can be formulated as follows:

$$\mathrm{MS} = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\left[\left(\frac{\max_{i\in \mathrm{NDS}} f_m^i - \min_{i\in \mathrm{NDS}} f_m^i}{F_m^{\max} - F_m^{\min}}\right)\right]^2}, \tag{15}$$

where M is the number of objectives, f_m^i is the function value of mth-objective function, $F_m^{\max}$ is the maximum value of mth-objective function in the true Pareto-front, and $F_m^{\min}$ is the minimum value of mth-objective function in the true Pareto-front. The larger the value of MS, the better is the obtained Pareto-optimal set of the solutions.

(c) Spacing Metric (SP)

SP provides a measure of uniformity of the spread among the obtained Pareto-optimal solutions, as introduced by Schott [47].

$$\mathrm{SP} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(\bar{d} - d_i\right)^2}, \tag{16}$$

where n is the number of Pareto-optimal solutions obtained, $\bar{d}$ is the average of all d_i, and$d_i = \min_{j \in \mathrm{NDS} \wedge j \neq i} \sum_{m=1}^{M} \left| f_m^i - f_m^j \right|$ and f_m^i is the function value of mth-objective function for solution i. The lower the value of SP, the better is the Pareto-optimal front of solutions.

5 Benchmark test problems

To assess the performances of the three proposed algorithms and compare them with that of other MOEAs, thirty test problems including CEC'09 multi-objective problems were chosen. These problems include UF1–UF10, ZDT1–ZDT6, SCH1–SCH2, "FON," "VIE," "POL," "KUR," "DEB," bi-objective DTLZ1, bi-objective DTLZ2, DTLZ3, DTLZ4, bi-objective DTLZ7, tri-objective DTLZ1, tri-objective DTLZ2, and tri-objective DTLZ3 (refer to Table 1). These test functions are of the different attributes and very challenging to solve. Each experiment with an algorithm was carried out for 25 times on each test problem.

6 Results and discussion

All the test problems were solved using the algorithms for 25 times, and the performance indices, such as inverse generation distance (IGD), maximum spread (MS), and spacing metric (SP), were calculated. Here, it is to be noted that for the calculations of IGD and MS, true Pareto-front (PF) is needed. The information regarding that true PFs are available for functions F02 through F14 and F26 through F30. However, the true PFs for the functions F01 and F15 through F25 are not available with the authors. Therefore, these two performance indices were determined only for the said 18 test functions. In case of SP, there is no need to get the information of the true PF. Hence, SP can be calculated for all the test functions. Besides these, in such cases, where the obtained PFs were found to have only one unique point each, all the three performance indices could not be determined.

6.1 Parameter settings

The performances of the proposed algorithms, namely MOBO-I, MOBO-II, and MOBO-III, are compared to four other efficient algorithms, such as Non-dominated Sorting Genetic Algorithm (NSGA-II), Multi-Objective Particle Swarm Optimization (MOPSO), Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) and Multi-Objective Grey-Wolf Optimizer (MOGWO). The MATLAB codes for MOPSO, MOEA/D were downloaded from: https://yarpiz.com/, whereas the code of MOGWO was obtained from: http://www.alimirjalili.com/GWO.html. Each algorithm was run up to 200 iterations. The parameters of the algorithms used for the purpose of comparison were selected through the parametric study, as suggested in [60]. In this method of parametric study, only one parameter was varied at a time and others were kept fixed. The selected parameters, which were found to yield the

Table 1 List of thirty multi-objective test problems

Function no./function name	No. of objectives/no. of variables	Properties of Pareto-front (PF)	Proposed by	Mathematical expressions available at
F01/'Kur'	2/3	Discontinuous	Kursawe [48]	[48–50]
F02/'Deb'	2/2	Continuous	Deb [51]	[49, 51]
F03/DTLZ1 (3 objectives)	3/12	Linear hyperplane	Deb et al. [52]	[50, 52, 53]
F04/DTLZ7 (3 objectives)	3/12	Discontinuous	Deb et al. [52]	[50, 52]
F05–F11/UF1–UF7 (2 objectives)	2/30	Convex, non-convex, discontinuous, and multimodal	Zhang et al. [54]	[10, 50, 54]
F12–F14/UF8–UF10 (3 objectives)	3/30		Zhang et al. [54]	[10, 54]
F15/ZDT1	2/30	Convex	Zitzler et al. [55]	[50, 53, 55]
F16/ZDT2	2/30	Non-convex	Zitzler et al. [55]	[50, 53, 55]
F17/ZDT3	2/30	Convex, discontinuous	Zitzler et al. [55]	[50, 53, 55]
F18/ZDT4	2/30	Non-convex	Zitzler et al. [55]	[50, 53, 55]
F19/ZDT6	2/30	Non-convex	Zitzler et al. [55]	[50, 53, 55]
F20/DTLZ2 (3 objectives)	3/7	Unit sphere	Deb et al. [52]	[50, 52, 53]
F21/DTLZ1 (2 objectives)	2/12	Continuous	Deb et al. [52]	[50, 52, 53]
F22/DTLZ2 (2 objectives)	2/10	Non-convex	Deb et al. [52]	[50, 52, 53]
F23/DTLZ3	2/12	Non-convex and has many local optimals	Deb et al. [52]	[50, 52, 53]
F24/DTLZ4	2/12	Non-convex	Deb et al. [52]	[50, 52]
F25/DTLZ7 (2 objectives)	2/12	Continuous	Deb et al. [52]	[50, 52]
F26/SCH1	2/1	Convex	Schaffer [56]	[53, 56]
F27/SCH2	2/1	Convex, discontinuous	Schaffer [56]	[53, 56]
F28/'FON'	2/3	Non-convex	Fonseca and Fleming [57]	[50, 53, 57]
F29/'POL'	2/2	Non-convex, discontinuous	Poloni [58]	[50, 58]
F30/'VIE'	3/2	Discontinuous	Viennet et al. [59]	[50, 59]

better results for most of the test functions, are reported in Table 2. Here, one point is to be noted that the authors are not claiming that these selected parameters are the best ones for solving all the test functions. However, these parameters were found to perform the best in most of the cases.

Table 3 provides the experimental mean values along with the standard deviation (SD) of the obtained SP values of 25 runs for all the seven multi-objective algorithms used for the comparisons. For the test functions involving three objectives, like F03 through F04,

Table 2 Parameter settings of the multi-objective optimization algorithms

Algorithm	Parameter	Numerical value
MOBO-I	Sharing coefficient for alpha-bonobo (scab)	1.55
	Sharing coefficient for selected bonobo (scsb)	1.4
	rate of change in phase probability (rcpp)	0.004
	Initial probability of extra-group mating ($p_{\text{xgm\_initial}}$)	0.08
	Maximum value for temporary subgroup size factor ($\text{tsgs}_{\text{factor\_max}}$)	0.07
	Number of grids per dimension	7
	Grid inflation rate	0.1
	Leader selection pressure	2
	Deletion selection pressure	2
MOBO-II	scab, scsb, rcpp, $p_{\text{xgm\_initial}}$, $\text{tsgs}_{\text{factor\_max}}$ are kept same as MOBO-I	
MOBO-III	scab, scsb, rcpp, $p_{\text{xgm\_initial}}$, $\text{tsgs}_{\text{factor\_max}}$ are kept same as MOBO-I	
	Repository size	100
	Number of neighbors	15
NSGA-II	Probability of crossover (p_{c})	0.9
	Probability of mutation (p_{m})	0.5
	Distribution index for simulated binary crossover (η_{c})	10
	Distribution index for polynomial mutation (η_{m})	20
MOPSO	Repository size	100
	Inertia weight (w)	0.5
	Inertia weight damping rate (wdamp)	0.99
	Personal learning coefficient (c_1)	1
	Global learning coefficient (c_2)	2
	Number of grids per dimension	10
	Grid inflation rate	0.1
	Leader selection pressure	2
	Deletion selection pressure	2
	Mutation rate	0.1
MOEA/D	Repository size	100
	Number of neighbors	15
	Crossover parameter (γ)	0.5
MOGWO	Repository size	100
	Grid inflation rate	0.1
	Number of grids per dimension	10
	Leader selection pressure	4
	Deletion selection pressure	2

F12 through F14, F20, and F30, the results yielded using MOEA/D and MOBO-III are not included, since the decomposition technique had not been developed in MATLAB [10]. For a few functions, such as F04, F16, and F25, MOGWO had produced a single unique solution in the repository. Similarly, MOPSO was also found to generate a single unique solution in the repository for the functions: F18 and F24. Therefore, SP values for these cases are not reported in Table 3. It is to be noted that the lower the SP value, the better is the performance of the

Table 3 Mean and standard deviation (SD) of the obtained SP values

Function		MOPSO	MOEA/D	MOGWO	NSGA-II	MOBO-I	MOBO-II	MOBO-III
F01	Mean	0.0634	0.4019	0.0902	0.0795	0.0642	0.0464	0.1660
	SD	0.0065	0.5016	0.0166	0.0047	0.0128	0.0078	0.0682
F02	Mean	0.0461	0.6300	0.0755	0.0287	0.0459	0.0364	0.1230
	SD	0.0056	0.4990	0.0218	0.0030	0.0056	0.0123	0.0547
F03	Mean	0.0221	–	0.1649	0.1537	0.0455	0.0257	–
	SD	0.0023	–	0.1166	0.0846	0.0103	0.0026	–
F04	Mean	0.0401	–	–	0.0298	0.0415	0.0524	–
	SD	0.0039	–	–	0.0064	0.0030	0.0047	–
F05	Mean	0.0164	0.0857	0.0251	0.0320	0.0972	0.0396	0.0570
	SD	0.0061	0.1097	0.0355	0.0242	0.0585	0.0180	0.0326
F06	Mean	0.0188	0.0229	0.0151	0.0129	0.0802	0.0687	0.1022
	SD	0.0047	0.0142	0.0045	0.0099	0.0286	0.0357	0.0386
F07	Mean	0.1069	0.0092	0.0959	0.0381	0.3286	0.0880	0.1280
	SD	0.0417	0.0044	0.0283	0.0264	0.2964	0.0833	0.1349
F08	Mean	0.0099	0.0394	0.0256	0.0061	0.0111	0.0059	0.0140
	SD	0.0011	0.0171	0.0103	0.0016	0.0016	0.0008	0.0028
F09	Mean	0.0634	0.0397	0.2958	0.2433	0.2024	0.1359	0.2433
	SD	0.0438	0.0842	0.3675	0.2247	0.1957	0.0696	0.1294
F10	Mean	0.0223	0.1488	0.1052	0.0781	0.2629	0.1487	0.1957
	SD	0.0172	0.3430	0.1632	0.0467	0.1558	0.1337	0.1532
F11	Mean	0.0120	0.0565	0.0132	0.0089	0.0882	0.0260	0.0532
	SD	0.0044	0.1008	0.0130	0.0050	0.0742	0.0188	0.0411
F12	Mean	0.3274	–	0.0280	0.0915	0.9830	0.3716	–
	SD	0.0848	–	0.0078	0.0336	0.2916	0.1224	–
F13	Mean	0.3829	–	0.0468	0.1913	0.9470	0.4272	–
	SD	0.1195	–	0.0130	0.0526	0.2761	0.1233	–
F14	Mean	1.2968	–	0.2954	0.4434	3.9605	1.6476	–
	SD	0.3872	–	0.0796	0.1109	1.1845	0.4920	–
F15	Mean	0.0140	0.0218	0.0204	0.0097	0.0103	0.0120	0.0139
	SD	0.0018	0.0070	0.0092	0.0007	0.0011	0.0032	0.0040
F16	Mean	0.0119	0.0277	–	0.0103	0.0097	0.0140	0.0108
	SD	0.0012	0.0162	–	0.0015	0.0010	0.0037	0.0020
F17	Mean	0.0109	0.0450	0.0179	0.0043	0.0109	0.0040	0.0111
	SD	0.0013	0.0187	0.0064	0.0003	0.0017	0.0003	0.0012
F18	Mean	–	2.4185	1.1639	0.0102	0.0609	0.0661	0.3182
	SD	–	5.3999	1.1283	0.0009	0.0444	0.0540	0.3451
F19	Mean	0.0090	0.1410	0.0409	0.0044	0.0430	0.0085	0.0332
	SD	0.0034	0.2630	0.0436	0.0007	0.0722	0.0037	0.0802
F20	Mean	0.0524	–	0.0185	0.0206	0.0576	0.0505	–
	SD	0.0045	–	0.0038	0.0039	0.0041	0.0051	–

Table 3 continued

Function		MOPSO	MOEA/D	MOGWO	NSGA-II	MOBO-I	MOBO-II	MOBO-III
F21	Mean	0.0053	0.0096	0.0477	0.0196	0.0135	0.0021	0.0056
	SD	0.0006	0.0032	0.0208	0.0042	0.0027	0.0004	0.0012
F22	Mean	0.0099	0.0143	0.0217	0.0116	0.0196	0.0128	0.0137
	SD	0.0011	0.0020	0.0076	0.0013	0.0049	0.0031	0.0028
F23	Mean	0.0098	0.0203	0.0816	0.0725	0.0524	0.0100	0.0146
	SD	0.0011	0.0111	0.0309	0.0312	0.0140	0.0019	0.0025
F24	Mean	–	0.0167	0.0576	0.0099	0.0402	0.0114	0.0121
	SD	–	0.0194	0.0667	0.0016	0.0169	0.0013	0.0038
F25	Mean	0.0124	0.0658	–	0.0047	0.0120	0.0096	0.0117
	SD	0.0015	0.0284	–	0.0003	0.0013	0.0013	0.0015
F26	Mean	0.0438	0.0899	0.0525	0.0790	0.0428	0.0827	0.0734
	SD	0.0038	0.0263	0.0168	0.0204	0.0049	0.0219	0.0242
F27	Mean	0.0668	0.3625	0.0900	0.0604	0.0703	0.0658	0.4755
	SD	0.0096	0.3772	0.0322	0.0153	0.0131	0.0205	0.5494
F28	Mean	0.0096	0.0146	0.0117	0.0814	0.0089	0.0078	0.0095
	SD	0.0009	0.0030	0.0022	0.0321	0.0009	0.0007	0.0012
F29	Mean	0.1262	0.5640	0.2304	0.1535	0.1558	0.1794	0.2211
	SD	0.0168	0.4505	0.0974	0.0377	0.0687	0.0560	0.3918
F30	Mean	0.0664	–	0.0929	0.0563	0.0758	0.0768	–
	SD	0.0084	–	0.0247	0.0259	0.0138	0.0158	–

algorithm. MOBO-II outperformed all other algorithms for F01, F17, F21, and F28, whereas MOBO-I was found to give the best SP values for F16 and F26. In this regard, NSGA-II and MOPSO were also observed to yield the good results in this experiment. However, among the proposed three versions of MOBO, MOBO-II had performed the better, and for most of the cases, the results obtained using MOBO-II were found to be comparable to that of the winner ones. Since SP is a metric used for uniformity of the spread, from the experimental results, it can be derived that the proposed MOBO-II has the better uniformity and divergence on average.

Table 4 provides the experimental mean and standard deviation results of IGD values using all the seven algorithms for 18 functions (as for the rest 12 functions, true PFs were not available with the authors). MOEA/D and MOBO-III had been excluded in the comparison for three objective functions. MOBO-II performed better than all other six algorithms for F03, F04, F08, and F28 functions. For the function: F26, MOBO-I was observed to provide the better IGD value. However, MOBO-III had the average performance in this regard. In this experiment, NSGA-II had performed well in terms of IGD values. IGD is the performance indicator for the convergence to the true Pareto-front. Therefore, MOBO-II had the best convergence ability to the true PF among the three proposed MOBOs.

The obtained MS values for the 18 functions are reported in Table 5 with their corresponding mean and SD values. MS value indicates the spread or coverage of the obtained Pareto-front with respect to the true one. In the experiment, MOBO-II had performed the best for the eight test functions, such as F05, F07 through F09, F14, and F26 through F28. For the function F03, MOBO-I had performed the best along with MOPSO and NSGA-II.

Table 4 Mean and standard deviation (SD) of the obtained IGD values

Function		MOPSO	MOEA/D	MOGWO	NSGA-II	MOBO-I	MOBO-II	MOBO-III
F02	Mean	0.0316	0.3059	0.0414	0.0206	0.0309	0.0484	0.0445
	SD	0.0034	0.1247	0.0062	0.0014	0.0031	0.0659	0.0157
F03	Mean	0.0425	–	0.1703	0.0746	0.1219	0.0344	–
	SD	0.0023	–	0.1363	0.0071	0.0299	0.0024	–
F04	Mean	4.1815	–	–	4.2559	4.1788	4.1737	–
	SD	0.0065	–	–	0.0366	0.0039	0.0000	–
F05	Mean	0.1036	0.3433	0.1317	0.0850	0.2257	0.1050	0.1796
	SD	0.0055	0.1906	0.0314	0.0135	0.0299	0.0033	0.0256
F06	Mean	0.1201	0.1327	0.0760	0.0577	0.5430	0.2882	0.5195
	SD	0.0167	0.0585	0.0077	0.0036	0.0422	0.0332	0.0610
F07	Mean	0.3008	0.4770	0.2948	0.4111	0.3850	0.3773	0.5492
	SD	0.0415	0.0888	0.0484	0.0079	0.0447	0.0131	0.0365
F08	Mean	0.0772	0.1165	0.0645	0.0867	0.0625	0.0491	0.0830
	SD	0.0049	0.0095	0.0050	0.0021	0.0039	0.0014	0.0026
F09	Mean	0.6306	2.8670	1.3077	1.0352	1.3555	0.8322	1.5984
	SD	0.1445	0.7014	0.5678	0.1501	0.1983	0.1045	0.1653
F10	Mean	0.5704	2.0589	0.4215	0.5108	0.7956	0.4954	0.8034
	SD	0.1192	0.7275	0.0196	0.0466	0.1178	0.0160	0.1284
F11	Mean	0.0697	0.4814	0.0896	0.0550	0.2079	0.0752	0.1724
	SD	0.0075	0.2374	0.0651	0.0254	0.0314	0.0037	0.0870
F12	Mean	0.7642	–	0.7439	0.2473	1.2669	0.6917	–
	SD	0.2323	–	0.3049	0.0110	0.4735	0.5382	–
F13	Mean	0.9023	–	0.1884	0.4036	1.5799	1.0545	–
	SD	0.2104	–	0.0449	0.0192	0.4950	0.3897	–
F14	Mean	5.7693	–	2.7583	2.6154	8.6364	5.9124	–
	SD	1.1056	–	0.5503	0.1854	2.6482	2.8128	–
F26	Mean	0.0358	0.1835	0.0500	0.0426	0.0349	0.0431	0.0601
	SD	0.0033	0.0617	0.0085	0.0131	0.0029	0.0127	0.0155
F27	Mean	0.0501	2.5319	0.0752	0.0454	0.0491	0.0495	2.6082
	SD	0.0054	2.5811	0.0266	0.0101	0.0068	0.0138	2.3459
F28	Mean	0.0069	0.0162	0.0188	0.0899	0.0069	0.0048	0.0077
	SD	0.0007	0.0063	0.0061	0.0322	0.0007	0.0005	0.0008
F29	Mean	0.1053	3.1532	0.1785	0.1345	0.1163	0.1327	9.7146
	SD	0.0109	4.0542	0.0513	0.0281	0.0189	0.0379	1.8360
F30	Mean	0.0605	–	0.0939	0.8619	0.0741	0.1013	–
	SD	0.0067	–	0.0163	0.3383	0.0079	0.0268	–

Table 5 Mean and standard deviation (SD) of the obtained MS values

Function		MOPSO	MOEA/D	MOGWO	NSGA-II	MOBO-I	MOBO-II	MOBO-III
F02	Mean	0.9982	0.8889	0.9847	0.9982	0.9981	0.9953	0.9855
	SD	0.0000	0.1207	0.0208	0.0000	0.0003	0.0097	0.0323
F03	Mean	1.0000	–	0.9946	1.0000	1.0000	0.9837	–
	SD	0.0000	–	0.0177	0.0000	0.0000	0.0205	–
F04	Mean	0.9361	–	–	0.8684	0.9356	0.9351	–
	SD	0.0012	–	–	0.0592	0.0005	0.0000	–
F05	Mean	0.9367	0.4652	0.9237	0.9242	0.9369	0.9872	0.9225
	SD	0.1144	0.1882	0.1117	0.0851	0.0575	0.0314	0.0779
F06	Mean	0.9784	0.6752	0.9839	0.9712	0.8092	0.9491	0.8133
	SD	0.0203	0.1679	0.0171	0.0180	0.0383	0.0162	0.0279
F07	Mean	1.0000	0.1963	0.9993	0.7377	0.9903	1.0000	0.5771
	SD	0.0000	0.0879	0.0025	0.0338	0.0473	0.0000	0.2378
F08	Mean	0.9690	0.8910	0.9695	0.9698	0.9689	0.9750	0.9719
	SD	0.0040	0.0407	0.0038	0.0014	0.0032	0.0009	0.0016
F09	Mean	0.5772	–	0.4166	0.5897	0.4113	0.7407	0.3513
	SD	0.2811	–	0.1817	0.1183	0.1156	0.0455	0.1330
F10	Mean	0.4871	0.8253	0.7155	0.5838	0.5682	0.7402	0.4726
	SD	0.2655	0.4414	0.0185	0.1339	0.1481	0.0208	0.1492
F11	Mean	0.9970	0.3881	0.9593	0.9667	0.9693	0.9924	0.9002
	SD	0.0014	0.2864	0.1406	0.0849	0.0070	0.0022	0.1567
F12	Mean	0.9558	–	0.4541	0.9722	0.8755	0.9597	–
	SD	0.0188	–	0.1630	0.0309	0.0715	0.0379	–
F13	Mean	0.9720	–	0.9152	0.9542	0.8551	0.9544	–
	SD	0.0204	–	0.1021	0.0290	0.1045	0.0241	–
F14	Mean	0.4392	–	0.4548	0.4802	–	0.7218	–
	SD	0.1117	–	0.1552	0.0516	–	0.2418	–
F26	Mean	0.9979	0.7428	0.9690	0.9999	0.9961	1.0000	0.9436
	SD	0.0018	0.0698	0.0125	0.0001	0.0032	0.0000	0.0481
F27	Mean	0.9968	0.6282	0.9745	0.9999	0.9939	0.9999	0.6635
	SD	0.0034	0.2801	0.0185	0.0001	0.0067	0.0001	0.2637
F28	Mean	0.9975	0.9420	0.9079	0.8880	0.9947	1.0000	0.9921
	SD	0.0025	0.0279	0.0315	0.0723	0.0043	0.0001	0.0073
F29	Mean	0.9969	0.7589	0.9897	0.9981	0.9959	0.9971	0.5822
	SD	0.0018	0.1643	0.0090	0.0016	0.0037	0.0036	0.0712
F30	Mean	0.9960	–	0.9518	0.8708	0.9916	0.9949	–
	SD	0.0029	–	0.0358	0.0206	0.0050	0.0039	–

For the functions: F01 through F04, both MOBO-I and MOBO-II along with MOPSO and NSGA-II had produced the low SP values. It showed that there was good uniformity in the spreads of the yielded solutions using these algorithms. MOBO-II had shown the best convergence among the algorithms for F04, whereas for F03, MOPSO had achieved the best convergence to the true PF. In terms of the MS values, both MOBO-I and MOBO-II had the good divergence. The results on benchmark function: F05 showed that the proposed MOBO-II had the better convergence, compared to MOEA/D and MOGWO in terms of IGD. However, the best convergence was obtained by NSGA-II for this function. In addition, MOBO-II outperformed the other two proposed algorithms in terms of IGD. MOBO-II and MOBO-III were able to outperform MOEA/D and MOBO-I with respect to the SP. However, they performed poorly, in comparison with MOPSO. In terms of MS, MOBO-II outperformed all the other algorithms followed by MOBO-I and MOPSO. Thus, on an average, MOBO-II and MOBO-I had shown both the good convergence and divergence for F05. However, MOBO-III had the better convergence, but poor divergence compared to the other two proposed algorithms for this function. The results on benchmark function: F06 showed that the proposed algorithms had poor convergence compared to other algorithms with MOBO-I being the worst. The best results were obtained by NSGA-II. With respect to SP, the proposed algorithms had poor performance on an average. Only MOBO-II had performed the better with respect to MS compared to the other proposed MOBOs. Thus, MOBO-II, MOBO-I and MOBO-III had poor convergence and divergence compared to other algorithms (except MOEA/D) for F06. The results on benchmark function F07 showed that MOBO-II outperformed all other algorithms except MOPSO and MOGWO with respect to IGD. In this case, MOBO-I and MOBO-III performed similar to most of the algorithms. With respect to SP, MOEA/D outperformed all the other algorithms, while MOBO-II performed better compared to the proposed algorithms. MOBO-I performed the worst out of all algorithms. On the other hand, MOBO-II and MOPSO were found to be the winners in terms of the obtained MS values. On an average, MOBO-II had the better convergence and superior divergence abilities for F07. For the functions F08 through F11, the proposed MOBO-II performed better in terms of both the convergence and divergence in comparison with other algorithms. However, the other proposed algorithms, like MOBO-I and MOBO-III, showed an average performance, in terms of both the qualities. For the three objective functions, such as F12 and F13, similar kind of performances was observed as discussed in previous cases. However, for the F14 function, all the proposed algorithms were not found to yield the good results. The proposed MOBO-II had yielded the best diverged PF for this problem among the algorithms for F14. As discussed earlier, only SP values had been reported for the functions: F15 through F25. For these cases, all the three MOBOs could yield good performance, except for the F18 using MOBO-III. For the rest of the functions, MOBO-I and MOBO-II had done well in terms of both the convergence and divergence properties of the obtained PFs. However, MOBO-III was seen to have the comparatively poor performance in these cases. Therefore, it had been observed in the experiment that the performances of the proposed algorithms were dependent on the nature of the functions.

As already mentioned, three indices were applied to determine the quantitative performances of the algorithms. For different functions, the algorithms had performed in different ways. Therefore, it is difficult to declare the overall winner algorithm in this experiment by mere observations. To solve this issue, nonparametric statistical tests [61] were applied to determine the relative ranks of the algorithms based on their performances. These statistical tests include Friedman, Aligned Friedman, and Quade tests for the multiple comparisons. To perform the tests, the mean results provided in Tables 3, 4, and 5 were considered. However, those results, which had incomplete information for all the algorithms, were not included in

Table 6 Relative ranks of the algorithms using three different statistical tests

Algorithm	Friedman	Aligned Friedman	Quade
MOPSO	2.5698	102.6860	2.4789
MOEA/D	5.7674	215.0233	5.6332
MOGWO	4.3023	122.7907	4.3235
NSGA-II	2.8605	122.0465	2.8319
MOBO-I	4.6047	181.3953	4.8784
MOBO-II	2.5233	103.2674	2.3985
MOBO-III	5.3721	209.7907	5.4556
p value	7.14E−11	2.52E−06	4.28E−19

Table 7 Contrast estimation results for the algorithms

	MOPSO	MOEA/D	MOGWO	NSGA-II	MOBO-I	MOBO-II	MOBO-III
MOPSO	0.0000	0.1030	0.0176	0.0100	0.0369	− 0.0006	0.0714
MOEA/D	− 0.1030	0.0000	− 0.0854	− 0.0930	− 0.0661	− 0.1036	− 0.0316
MOGWO	− 0.0176	0.0854	0.0000	− 0.0076	0.0193	− 0.0182	0.0538
NSGA-II	− 0.0100	0.0930	0.0076	0.0000	0.0269	− 0.0106	0.0614
MOBO-I	− 0.0369	0.0661	− 0.0193	− 0.0269	0.0000	− 0.0375	0.0345
MOBO-II	0.0006	0.1036	0.0182	0.0106	0.0375	0.0000	0.0720
MOBO-III	− 0.0714	0.0316	− 0.0538	− 0.0614	− 0.0345	− 0.0720	0.0000

this analysis. The obtained statistical results are reported in Table 6 for all the three tests. Here, the relative ranks are given for all the algorithms. The lower the rank, the better is its performance. From the results, it is obvious that MOBO-II is declared as the best algorithm by Friedman and Quade tests. Moreover, it is followed by MOPSO, NSGA-II, MOGWO, MOBO-I, and MOBO-III. MOEA/D is seen to be the worst performer in these tests. In case of Aligned Friedman test, MOPSO is found to have the lowest rank and MOBO-II is assumed to be the second best performer. The performances of rest of the algorithms are similar to that of Friedman and Quade tests. Besides these, the low p values (<0.5) of the tests suggest significant differences among the performances of the applied algorithms. It is to be noticed that in three tests, the difference in performances between MOBO-II and MOPSO was found to be very less. NSGA-II was also seen to yield the comparable performance compared to these two algorithms. However, the rest of the algorithms had performed distinctly worse in this experiment.

Apart from the relative ranks obtained using three different statistical techniques, contrast estimation results for the algorithms are also provided in Table 7. This test is going to measure the difference in performances between two techniques. In this table, if a measured value is observed to be a positive one, then the method present in the first column is regarded to have the better performance compared to that of the other columns. In addition, the more the value, the more will be the difference in performance. In Table 7, all the estimated values for MOBO-II are found to be positive. It shows that MOBO-II has the overall superior performance compared to rest of the algorithms and it is followed by MOPSO, NSGA-II, MOGWO, MOBO-I, MOBO-III, and MOEA/D.

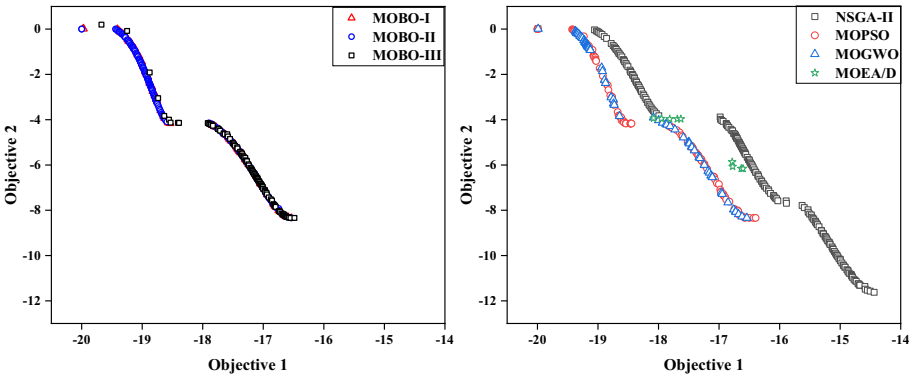

Fig. 7 Pareto-fronts of solutions obtained using proposed MOBOs and other four popular algorithms for the functions F01

In the above discussion, it has been observed that MOBO-II performs better than the other two proposed algorithms for most of the functions. For the qualitative comparisons, Figs. 7, 8, 9, 10, 11, and 12 display the obtained PFs for all the used algorithms in solving different test problems. From the figures, it is clear that in most of the cases, MOBO-II could yield the better PFs compared to the other two. Moreover, in Fig. 13, the box-plots of the three performance indices are shown for the functions: F05 and F08 (here, these two functions are chosen at random from the CEC'09 test suite). From these plots, it is clear that the proposed MOBO-II had the superior and consistent performances compared to other algorithms. These results strongly establish the fact that the proposed MOBO-II is an efficient algorithm in the domain of multi-objective optimization.

6.2 Time-study results

To compute the time required for each algorithm, all the test problems had been solved for five times, and the calculated average CPU times are reported in Table 8 test function-wise. Here, the problems with three objectives were excluded from the experiment, as the MATLAB codes of MOEA/D and MOBO-III were not able to solve those problems. This study was performed on the Windows 10 platform having 32 GB RAM and Intel i7 core processor. From the study, it was evident that both MOBO-II and NSGA-II had taken less CPU time compared to other algorithms for all the test functions. Moreover, MOBO-II required even less CPU time compared to NSGA-II. It means that MOBO-II was the fastest algorithm used in this study. Apart from these two, MOBO-I was also seen to take less CPU time followed by MOPSO, MOGWO, and MOBO-III. However, MOBO-III was found to take less CPU time compared to MOGWO in a few cases. MOEA/D was observed to be the most computationally expensive one, in this study. MOBO-II was also found to be a fast algorithm, like NSGA-II. On the other hand, MOPSO with the grid mechanism was found to be computationally expensive. Due to this reason, both MOBO-I and MOGWO were seen to be computationally more expensive. However, MOBO-I was found to be less expensive compared to MOPSO and MOGWO, as BO itself is less computationally expensive compared to PSO and GWO. In this experiment, MOEA/D was found to be the most computationally expensive algorithm. Due to this reason, MOBO-III was also seen to be computationally expensive, as it was developed based on the framework of MOEA/D.

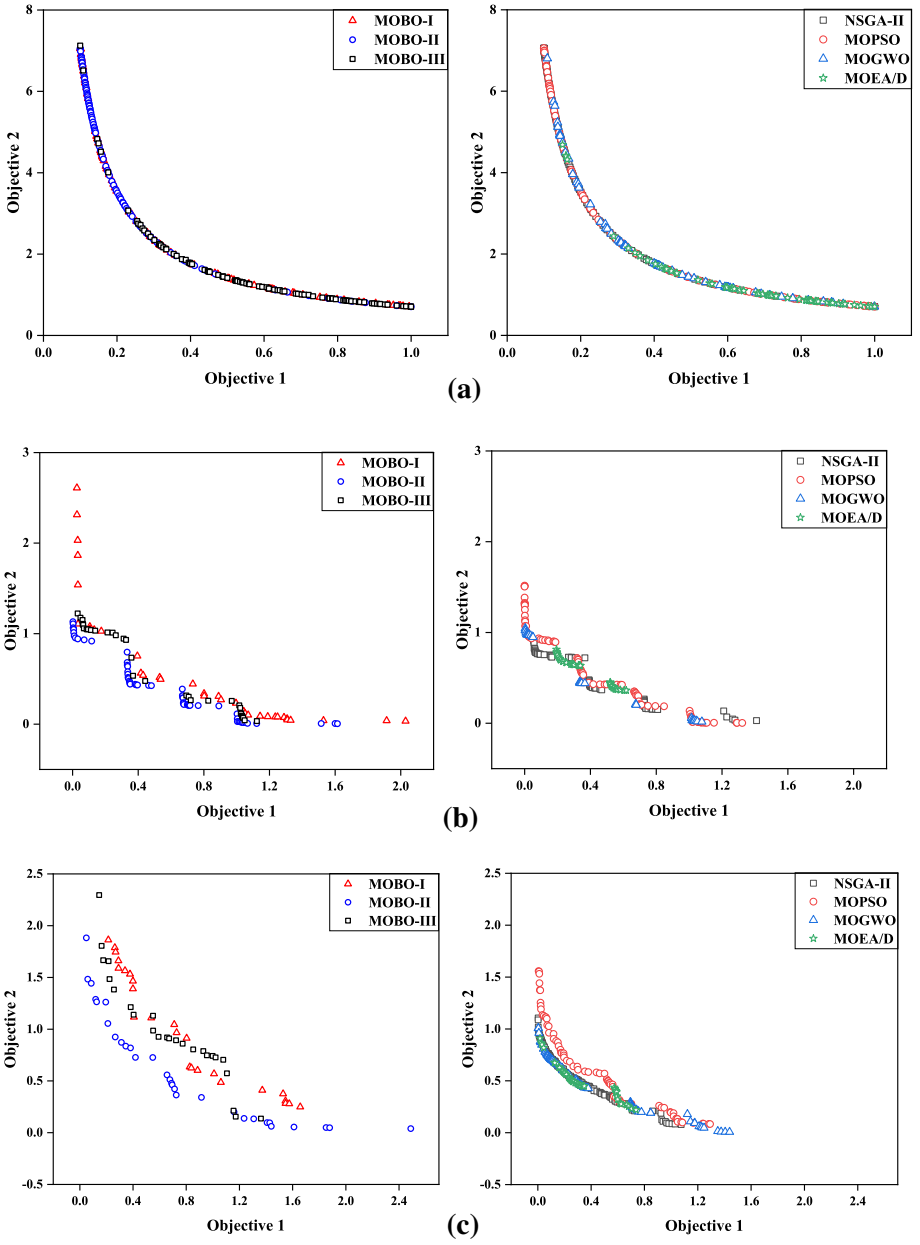

Fig. 8 Pareto-fronts of solutions obtained using proposed MOBOs and other four popular algorithms for the functions: **a** F02, **b** F05 and **c** F06

6.3 Analysis of results

The recently developed BO [21] was found to be an efficient algorithm due to its intelligent and adaptive search mechanism. The working principles and the goals for the single- and multi-

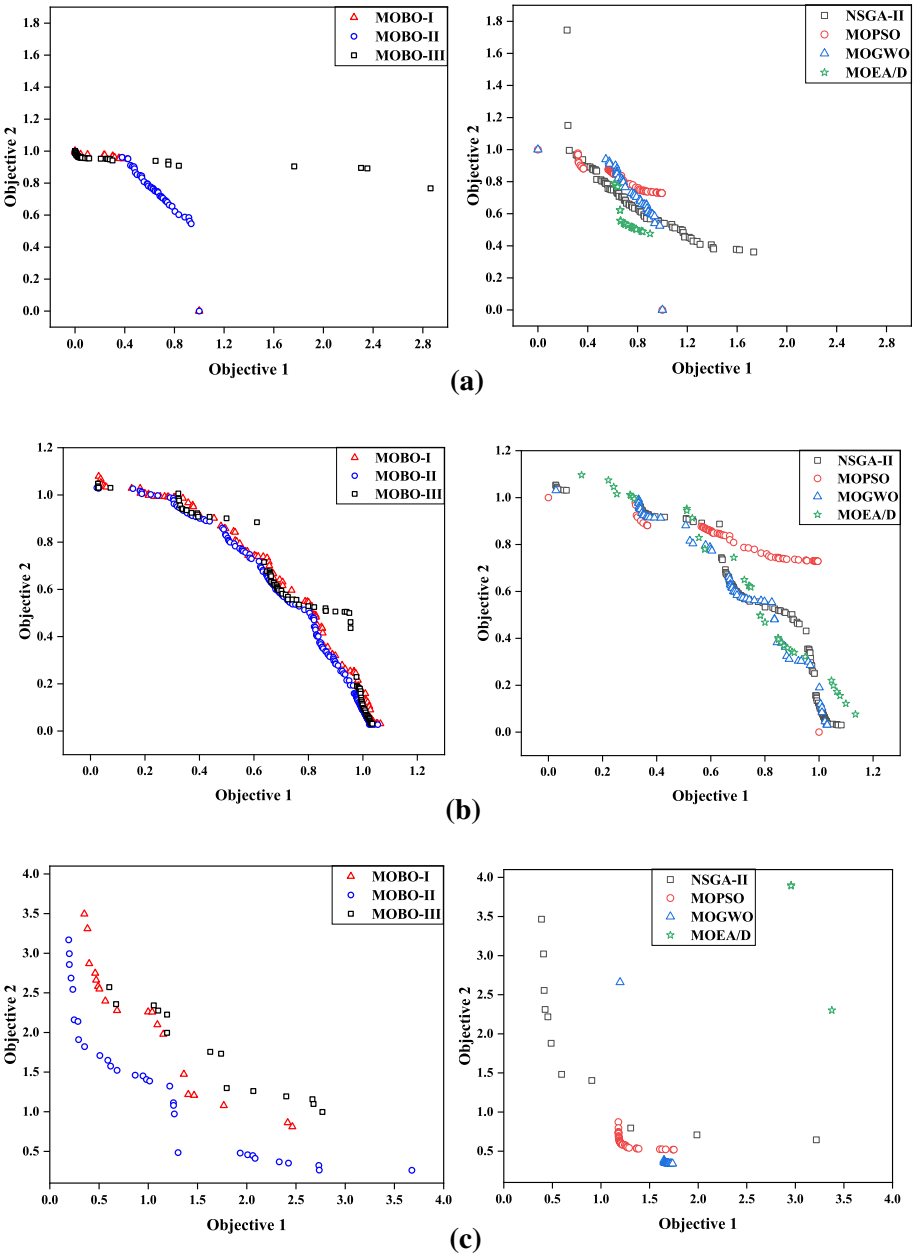

Fig. 9 Pareto-fronts of solutions obtained using proposed MOBOs and other four popular algorithms for the functions: **a** F07, **b** F08, and **c** F09

objective optimizations are different. Moreover, the performance criteria are found to be not similar in both the cases. In case of multi-objective optimization, the basic framework used for converting single to multi-objectives has significant role in improving the performance. In our experiment, the non-dominated sorting and crowding distance approaches had been

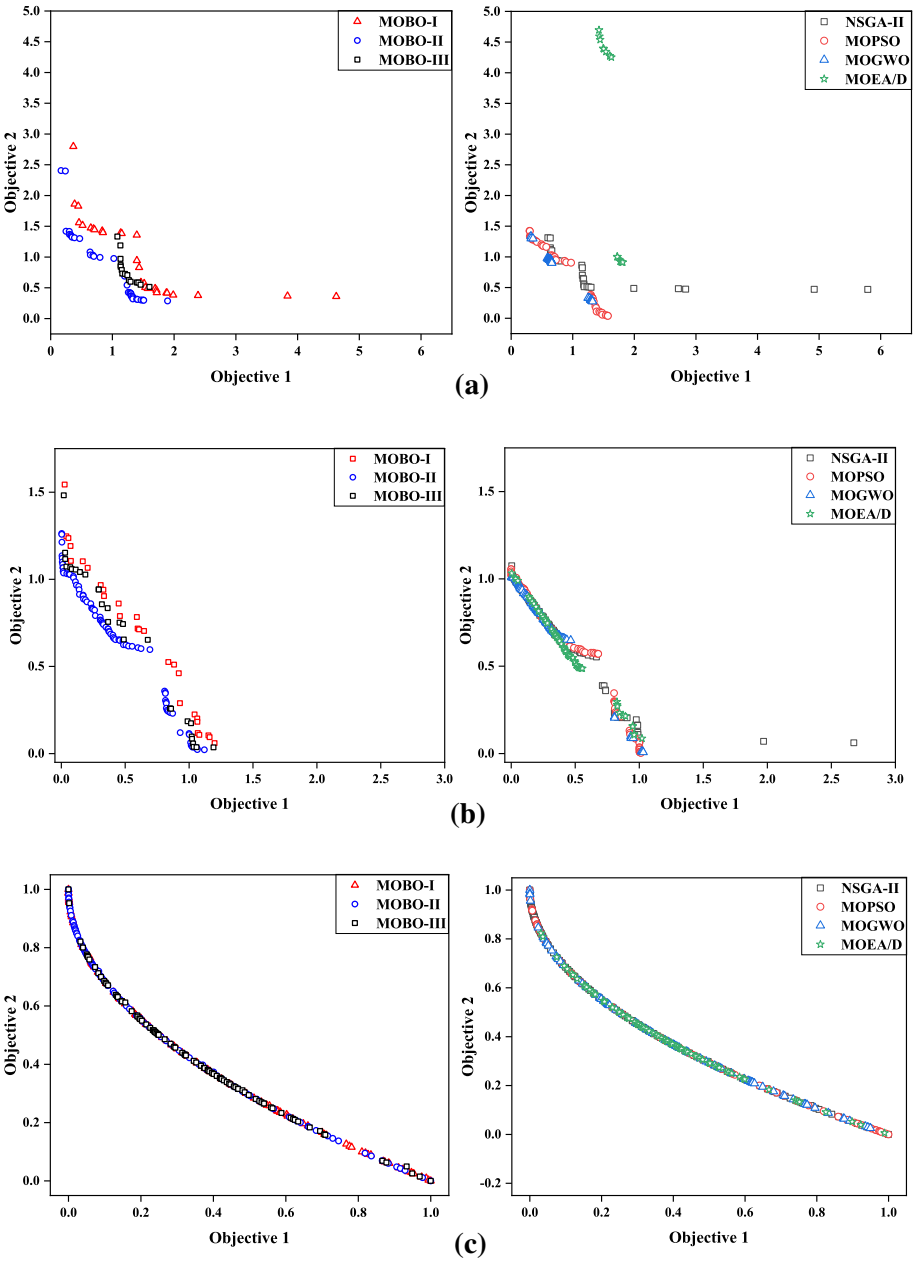

Fig. 10 Pareto-fronts of solutions obtained using proposed MOBOs and other four popular algorithms for the functions: **a** F10, **b** F11, and **c** F15

proved to be more suitable framework for converting the BO to MOBO. This may be due to the fact that in MOBO-II, all the solutions participate in the selection process of *p*th-bonobo. This increases diversity in the population. In case of MOBO-I, only the members of the repository can take part in the selection process. Apart from this, the acceptance criterion for the offspring may create the difference among the proposed algorithms. In MOBO-II, from the

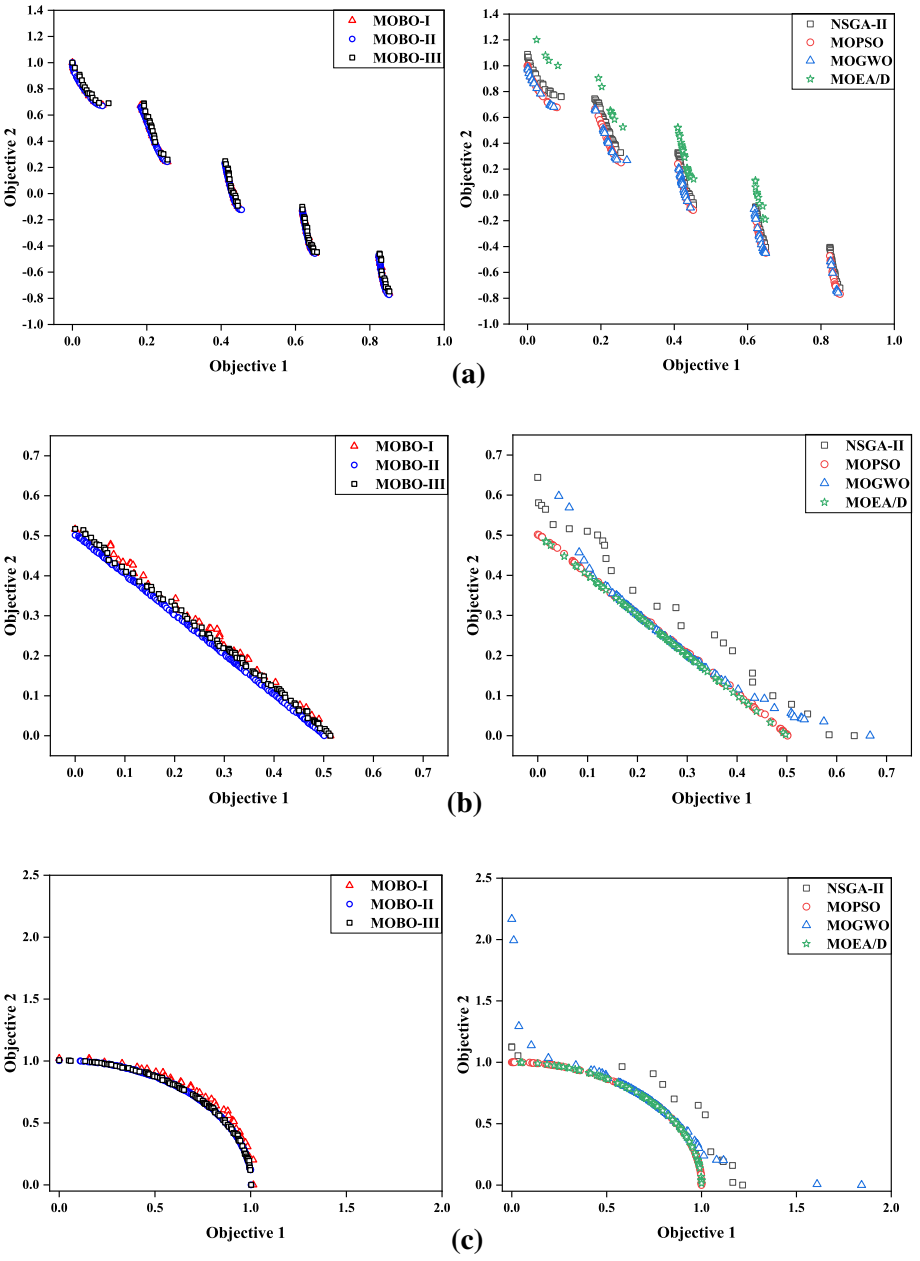

Fig. 11 Pareto-fronts of solutions obtained using proposed MOBOs and other four popular algorithms for the functions: **a** F17, **b** F21, and **c** F23

mixed population of the parent and children solutions, current population was selected based on the rankings and crowding distances. However, in MOBO-I, all the children solutions created through the positive phase were accepted directly and the solutions created through the negative phase were considered under certain conditions. Due to this reason, uniformity

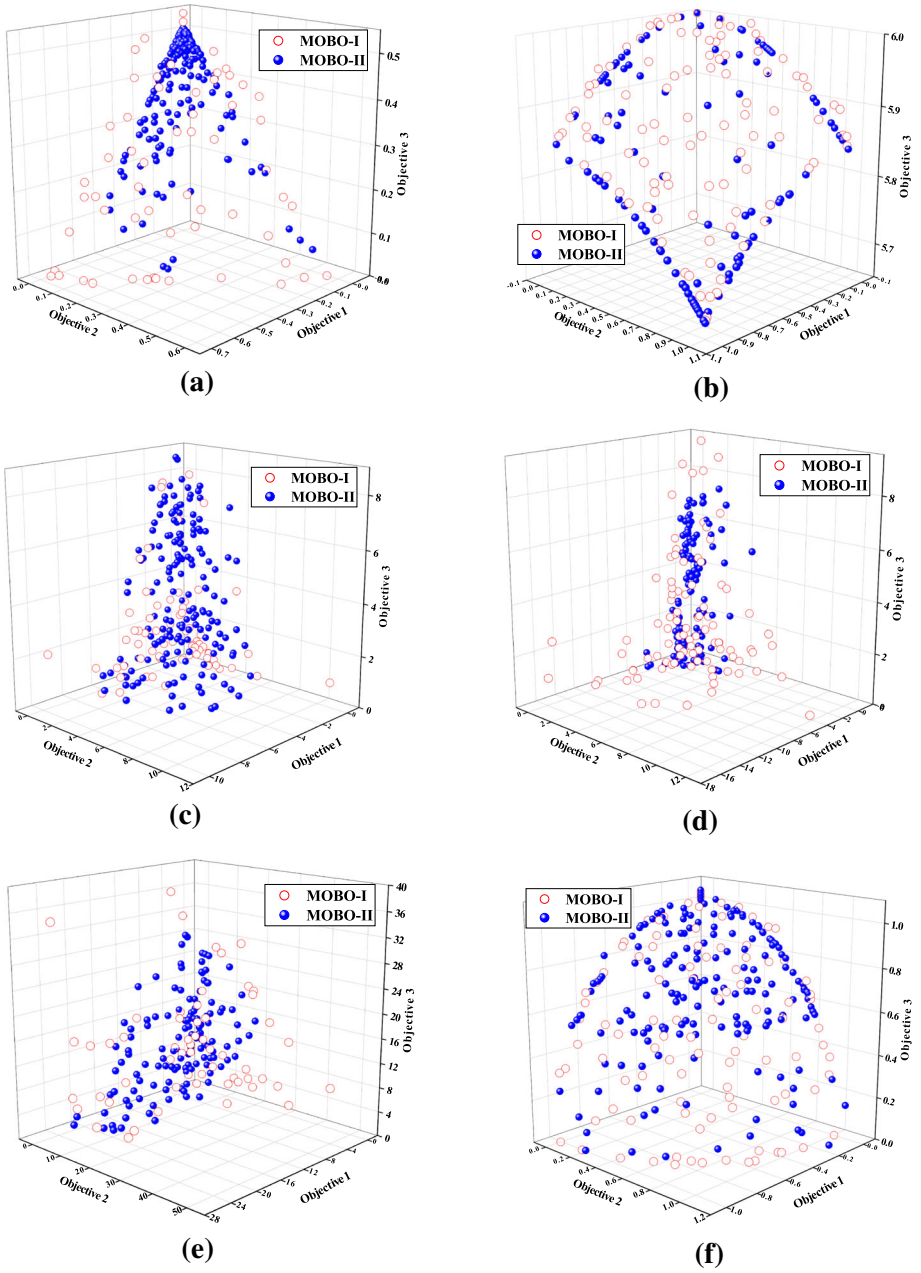

Fig. 12 Pareto-fronts of solutions obtained using proposed MOBOs for the functions: **a** F03, **b** F04, **c** F12, **d** F13, **e** F14, and **f** F20

and convergence to the true Pareto-front capabilities of MOBO-I were sometimes found to be poor, as each solution created in the positive phase was accepted. However, the divergence power of MOBO-I was found to be good for the same reason. The criterion used in MOBO-

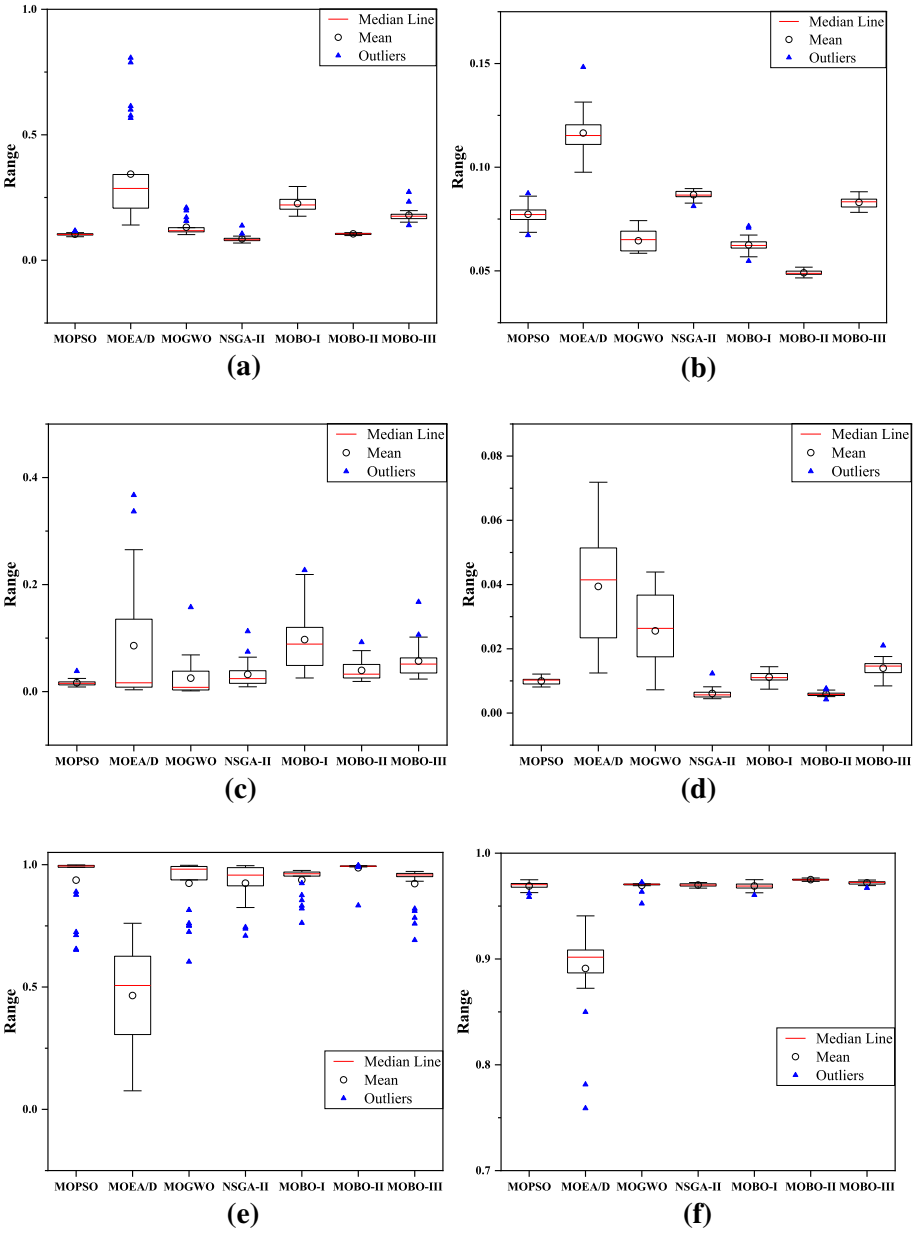

Fig. 13 Box plots of the performance indicators in 25 runs for all the algorithms: **a** IGD for F05, **b** IGD for F08, **c** SP for F05, **d** SP for F08, **e** MS for F05, and **f** MS for F08

II was found to be more efficient, as all the parent and children solutions had the chances to compete one another. It increases the probability of getting the more diversified Pareto-optimal solutions through the iterations. It maintains a proper balance between the divergence and convergence power of the algorithm. However, it is not the same in case of MOBO-I. On the other hand, MOBO-III was developed based on the decomposition technique. In the experiment, MOEA/D had not worked well and it was observed to be the worst performer

Table 8 CPU time-study results in seconds for the algorithms

Function	MOPSO	MOEA/D	MOGWO	NSGA-II	MOBO-I	MOBO-II	MOBO-III
F01	52.46	103.35	100.93	9.43	38.40	7.65	102.92
F02	87.18	140.79	83.26	8.95	42.09	7.89	103.94
F05	45.82	109.53	30.96	20.98	37.26	8.22	63.24
F06	44.98	132.08	95.59	23.36	36.66	8.50	65.64
F07	47.12	99.98	38.85	23.68	37.97	9.83	57.64
F08	56.80	139.75	80.33	22.40	46.02	8.88	97.78
F09	42.30	91.67	19.10	22.11	37.26	8.43	57.25
F10	43.46	94.94	22.14	23.57	37.92	8.88	57.95
F11	47.28	106.37	63.00	21.24	37.37	8.24	61.90
F15	126.31	144.10	201.79	10.02	58.06	8.36	161.20
F16	94.75	133.63	201.61	19.66	48.21	8.38	106.46
F17	57.10	107.94	113.19	19.75	42.39	8.02	76.45
F18	55.66	63.20	19.52	12.38	35.20	7.76	52.07
F19	110.48	121.13	117.49	11.92	48.52	8.23	131.28
F21	67.11	141.82	21.91	12.61	33.70	7.48	89.30
F22	73.14	144.16	93.37	11.96	33.89	7.36	123.57
F23	67.64	144.48	20.53	12.71	34.05	7.94	97.18
F24	59.16	126.03	45.70	12.71	33.99	7.54	78.60
F25	95.36	81.26	194.64	12.73	52.86	7.91	141.04
F26	95.45	145.43	186.21	9.10	60.18	7.92	164.12
F27	70.46	119.63	121.68	8.90	53.06	7.57	147.75
F28	55.82	145.14	147.66	8.95	42.50	7.12	94.57
F29	57.52	140.55	100.79	8.89	44.41	7.39	110.40

among all the algorithms. In a similar way, the performance of the MOBO-III was also not satisfactory and it was the second worst performer among the algorithms. Finally, it is concluded that MOBO-II was the best performer and the corresponding framework for converting BO to MOBO was seen to be the most suitable one. The performance of MOBO-I was found to be average and that of MOBO-III was the worst one. It is clear from the above discussion that all the proposed MOBOs were not equally potential to yield good results for several types of optimization problems. Moreover, MOBO-I and MOBO-III were found to be computationally expensive. It is a disadvantage for both the proposed MOBOs. MATLAB code of MOBO-III was not able to solve three objective optimization problems in its present form. However, the proposed MOBO-II was found to be both more efficient as well as computationally faster algorithm compared to others. Therefore, these are some of the pros and cons of the proposed MOBOs.

7 Concluding remarks

In this paper, three novel multi-objective optimization algorithms, which are also the multi-objective versions of the recently proposed single-objective Bonobo Optimizer (BO), had been proposed. BO was found to work exceptionally well compared to other popular algo-

rithms. However, the performance of proposed Multi-objective Bonobo Optimizer (MOBO) was found to be dependent not only on the performance of BO, but also on the framework or strategy used for converting BO into MOBO. In this study, three distinct approaches available in the literature had been used to develop these multi-objective versions of BO. The second version of MOBO (i.e., MOBO-II), which was developed based on the non-dominated sorting and crowding distance, had been found to perform the better compared to other two approaches, which were designed based on grid index and decomposition technique, respectively. The performance of MOBO-II was seen to be the superior in terms of both the convergence to true Pareto-front and divergence of the obtained Pareto-front. In addition, the results on the thirty challenging test problems revealed the supreme performance of the proposed MOBO-II among all other popular multi-objective algorithms, such as NSGA-II, MOPSO, MOGWO and MOEA/D. The performance of MOBO-I with the grid index approach was observed to be the average. However, MOBO-III with the decomposition technique had performed the worst in the experiment. From these observations, it is concluded that the multi-objective framework used to convert BO to MOBO has a great role to play in its overall performance. These proposed MOBOs will be applied in future to solve more number of test functions and real-world problems.

References

1. Marler RT, Arora JS (2004) Survey of multi-objective optimization methods for engineering. Struct Multidiscip Optim 26(6):369–395. https://doi.org/10.1007/s00158-003-0368-6
2. Ghiassi M, Devor RE, Dessouky MI, Kijowski BA (1984) An application of multiple criteria decision making principles for planning machining operations. IIE Trans 16(2):106–114. https://doi.org/10.1080/07408178408974675
3. Fox AD, Corne DW, Mayorga Adame CG, Polton JA, Henry L-A, Roberts JM (2019) An efficient multi-objective optimization method for use in the design of marine protected area networks. Front Mar Sci. https://doi.org/10.3389/fmars.2019.00017
4. Acharya PS (2019) Intelligent algorithmic multi-objective optimization for renewable energy system generation and integration problems: a review. Int J Renew Energy Res 9(1):271–280
5. Gopakumar AM, Balachandran PV, Xue D, Gubernatis JE, Lookman T (2018) Multi-objective optimization for materials discovery via adaptive design. Sci Rep 8(1):3738
6. Franken T, Duggan A, Matrisciano A, Lehtiniemi H, Borg A, Mauss F (2019) Multi-objective optimization of fuel consumption and NOx emissions with reliability analysis using a stochastic reactor model. SAE technical paper, 2019-01-1173. https://doi.org/10.4271/2019-01-1173
7. 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. https://doi.org/10.1109/4235.996017
8. Zhang Q, Li H (2007) MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11(6):712–731. https://doi.org/10.1109/TEVC.2007.892759
9. Coello CAC, Lechuga MS (2002) MOPSO: a proposal for multiple objective particle swarm optimization. In: Proceedings of the congress on evolutionary computation (CEC'02), pp 1051–1056. https://doi.org/10.1109/cec.2002.1004388
10. Mirjalili S, Saremi S, Mirjalili SM, Coelho LdS (2016) Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. Expert Syst Appl 47:106–119. https://doi.org/10.1016/j.eswa.2015.10.039
11. 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. https://doi.org/10.1007/s10489-016-0825-8
12. Golberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Longman Publishing Co, Reading
13. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science (MHS'95). IEEE, pp 39–43
14. Shabani A, Asgarian B, Gharebaghi SA, Salido MA, Giret A (2019) A new optimization algorithm based on search and rescue operations. Math Probl Eng. https://doi.org/10.1155/2019/2482543

15. Harifi S, Khalilian M, Mohammadzadeh J, Ebrahimnejad S (2019) Emperor Penguins Colony: a new metaheuristic algorithm for optimization. Evol Intel 12(2):211–226. https://doi.org/10.1007/s12065-019-00212-x
16. Hayyolalam V, Pourhaji Kazem AA (2020) Black Widow Optimization Algorithm: a novel meta-heuristic approach for solving engineering optimization problems. Eng Appl Artif Intell 87:103249. https://doi.org/10.1016/j.engappai.2019.103249
17. Zervoudakis K, Tsafarakis S (2020) A mayfly optimization algorithm. Comput Ind Eng 145:106559. https://doi.org/10.1016/j.cie.2020.106559
18. Fathollahi-Fard AM, Hajiaghaei-Keshteli M, Tavakkoli-Moghaddam R (2020) Red deer algorithm (RDA): a new nature-inspired meta-heuristic. Soft Comput. https://doi.org/10.1007/s00500-020-04812-z
19. Dokeroglu T, Sevinc E, Kucukyilmaz T, Cosar A (2019) A survey on new generation metaheuristic algorithms. Comput Ind Eng 137:106040. https://doi.org/10.1016/j.cie.2019.106040
20. Wong W, Ming CI (2019) A review on metaheuristic algorithms: recent trends, benchmarking and applications. In: 2019 7th international conference on smart computing and communications (ICSCC), 28–30 June 2019, pp 1–5. https://doi.org/10.1109/icscc.2019.8843624
21. Das AK, Pratihar DK (2019) A new Bonobo Optimizer (BO) for real-parameter optimization. In: IEEE region 10 symposium (TENSYMP 2019) Kolkata, India, pp 108–113. https://doi.org/10.1109/tensymp46218.2019.8971108
22. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82. https://doi.org/10.1109/4235.585893
23. Deb K (2012) Advances in evolutionary multi-objective optimization. In: Fraser G, Teixeira de Souza J (eds) Search based software engineering (SSBSE), 2012. Lecture notes in computer science. Springer, Berlin, pp 1–26. https://doi.org/10.1007/978-3-642-33119-0_1
24. Zadeh L (1963) Optimality and non-scalar-valued performance criteria. IEEE Trans Autom Control 8(1):59–60. https://doi.org/10.1109/TAC.1963.1105511
25. 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. https://doi.org/10.1109/4235.797969
26. Bhargava S (2013) A note on evolutionary algorithms and its applications. Adults Learn Math 8(1):31–45
27. Knowles JD, Corne DW (2000) Approximating the nondominated front using the Pareto archived evolution strategy. Evol Comput 8(2):149–172. https://doi.org/10.1162/106365600568167
28. Srinivas N, Deb K (1994) Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evol Comput 2(3):221–248. https://doi.org/10.1162/evco.1994.2.3.221
29. Xiangui S, Dekui K (2015) A multi-objective ant colony optimization algorithm based on elitist selection strategy. Metall Min Ind 7(6):333–338
30. Jiang S, Yang S (2017) A strength Pareto evolutionary algorithm based on reference direction for multiobjective and many-objective optimization. IEEE Trans Evol Comput 21(3):329–346. https://doi.org/10.1109/TEVC.2016.2592479
31. Zeng J, Dou L, Xin B (2018) Multi-objective cooperative salvo attack against group target. J Syst Sci Complex 31(1):244–261. https://doi.org/10.1007/s11424-018-7437-9
32. Zapotecas-Martínez S, López-Jaimes A, García-Nájera A (2019) LIBEA: a Lebesgue indicator-based evolutionary algorithm for multi-objective optimization. Swarm Evol Comput 44:404–419. https://doi.org/10.1016/j.swevo.2018.05.004
33. Foroughi Nematollahi A, Rahiminejad A, Vahidi B (2019) A novel multi-objective optimization algorithm based on lightning attachment procedure optimization algorithm. Appl Soft Comput 75:404–427. https://doi.org/10.1016/j.asoc.2018.11.032
34. Liu C, Du Y, Li A, Lei J (2020) Evolutionary multi-objective membrane algorithm. IEEE Access 8:6020–6031. https://doi.org/10.1109/ACCESS.2019.2939217
35. RamuNaidu Y, Ojha AK, SusheelaDevi V (2020) Multi-objective Jaya Algorithm for solving constrained multi-objective optimization problems. In: Kim JH, Geem ZW, Jung D, Yoo DG, Yadav A (eds) Advances in Harmony search, soft computing and applications. Springer, Cham, pp 89–98
36. Han X, Liu J (2020) Micro multi-objective genetic algorithm. In: Han X, Liu J (eds) Numerical simulation-based design: theory and methods. Springer, Singapore, pp 153–178. https://doi.org/10.1007/978-981-10-3090-1_9
37. Wu D, Gao H (2020) Multi-objective bird swarm algorithm. In: Lu H (ed) Cognitive internet of things: frameworks, tools and applications. Springer, Cham, pp 109–119. https://doi.org/10.1007/978-3-030-04946-1_12
38. Gunantara N (2018) A review of multi-objective optimization: methods and its applications. Cogent Eng 5(1):1502242. https://doi.org/10.1080/23311916.2018.1502242
39. Emmerich MTM, Deutz AH (2018) A tutorial on multiobjective optimization: fundamentals and evolutionary methods. Nat Comput 17(3):585–609. https://doi.org/10.1007/s11047-018-9685-y

40. Huang W, Zhang Y, Li L (2019) Survey on multi-objective evolutionary algorithms. J Phys: Conf Ser 1288:012057. https://doi.org/10.1088/1742-6596/1288/1/012057
41. Ojstersek R, Brezocnik M, Buchmeister B (2020) Multi-objective optimization of production scheduling with evolutionary computation: a review. Int J Ind Eng Comput 11(3):359–376
42. Social Organization. http://luna.cas.usf.edu/~rtykot/ANT3101/primates/organization.html. Accessed 23/10/2019
43. Holland JH (1992) Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence. MIT Press, Cambridge
44. Miettinen K (2012) Nonlinear multiobjective optimization, vol 12. Springer, New York. https://doi.org/10.1007/978-1-4615-5563-6
45. Zitzler E, Thiele L (1998) Multiobjective optimization using evolutionary algorithms—a comparative case study. In: Eiben AE, Bäck T, Schoenauer M, Schwefel H-P (eds) Parallel problem solving from nature—PPSN V. Lecture notes in computer science. Springer, Berlin, pp 292–301. https://doi.org/10.1007/bfb0056872
46. Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, Chichester
47. Schott JR (1995) Fault tolerant design using single and multicriteria genetic algorithm optimization. Master thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Boston
48. Kursawe F (1991) A variant of evolution strategies for vector optimization. In: Schwefel HP, Männer R (eds) International conference on parallel problem solving from nature (PPSN). Lecture notes in computer science. Springer, Berlin, pp 193–197. https://doi.org/10.1007/bfb0029752
49. http://delta.cs.cinvestav.mx/~ccoello/EMOO/testfuncs/. Accessed on 23/09/2019
50. http://jmetal.sourceforge.net/problems.html. Accessed on 23/09/2019
51. Deb K (1999) Multi-objective genetic algorithms: problem difficulties and construction of test problems. Evol Comput 7(3):205–230. https://doi.org/10.1162/evco.1999.7.3.205
52. Deb K, Thiele L, Laumanns M, Zitzler E (2005) Scalable test problems for evolutionary multiobjective optimization. In: Abraham A, Jain L, Goldberg R (eds) Evolutionary multiobjective optimization. Advanced information and knowledge processing. Springer, London, pp 105–145. https://doi.org/10.1007/1-84628-137-7_6
53. Gong W, Duan Q, Li J, Wang C, Di Z, Ye A, Miao C, Dai Y (2016) Multiobjective adaptive surrogate modeling-based optimization for parameter estimation of large, complex geophysical models. Water Resour Res 52(3):1984–2008
54. 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
55. Zitzler E, Deb K, Thiele L (2000) Comparison of multiobjective evolutionary algorithms: empirical results. Evol Comput 8(2):173–195
56. Schaffer JD (1985) Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the first international conference on genetic algorithms and their applications. Lawrence Erlbaum Associates Inc., Publishers
57. Fonseca CM, Fleming PJ (1998) Multiobjective optimization and multiple constraint handling with evolutionary algorithms. I. A unified formulation. IEEE Trans Syst Man Cybern Part A Syst Hum 28(1):26–37
58. Poloni C, Giurgevich A, Onesti L, Pediroda V (2000) Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics. Comput Methods Appl Mech Eng 186(2–4):403–420
59. Viennet R, Fonteix C, Marc I (1996) Multicriteria optimization using a genetic algorithm for determining a Pareto set. Int J Syst Sci 27(2):255–260. https://doi.org/10.1080/00207729608929211
60. Pratihar DK (2013) Soft computing: fundamentals and applications. Alpha Science International Ltd, Oxford
61. García S, Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. J Heuristics 15(6):617

Mr. Amit Kumar Das received his B.E. and M.Tech. from Indian Institute of Engineering Science and Technology, Shibpur, West Bengal, India (formerly Bengal Engineering and Science University, Shibpur) and IIT Kharagpur, India, in 2008 and 2016, respectively. He is currently pursuing his Ph.D. at IIT Kharagpur, West Bengal, India. His research interests include intelligent optimization algorithm development and its applications to engineering, machine learning, robotics, etc.

Ankit Kumar Nikum is a senior undergraduate student in the Department of Mechanical Engineering at the Sardar Vallabhbhai National Institute of Technology, Surat. He will be graduating in 2021 with a B.Tech. in Mechanical Engineering. He has strong interests in the field of robotics, Machine learning, motion planning, and optimization Techniques.

Siva Vignesh Krishnan is a senior undergraduate student in the Department of Mechanical Engineering at the Indian Institute of Technology Kharagpur, India. He will be graduating in 2021 with a B.Tech. in Mechanical Engineering and a minor in Mathematics and Computing. He has strong interests in the field of robotics, control systems, motion planning, and convex optimization.

Dr. Dilip Kumar Pratihar received his B.E. (Hons.) and M.Tech. from REC (NIT) Durgapur, India, in 1988 and 1994, respectively. He obtained his Ph.D. from IIT Kanpur, India, in 2000. He received University Gold Medal, A.M. Das Memorial Medal, Institution of Engineers (I) Medal, and others. He completed his postdoctoral studies in Japan and then in Germany under the Alexander von Humboldt Fellowship Programme. He is working as a Professor of IIT Kharagpur, India. His research areas include robotics, soft computing and manufacturing science. He has published more than 270 papers, mostly in various international journals. He has written the textbooks on "Soft Computing," "Fundamentals of Robotics," co-authored another textbook on "Analytical Engineering Mechanics," edited a book on "Intelligent and Autonomous Systems," co-authored reference books on "Modeling and Analysis of Six-legged Robots," "Modeling and Simulations of Robotic Systems Using Soft Computing," "Modeling and Analysis of Laser Metal Forming Processes by Finite Element and Soft Computing," "Multi-body Dynamic Modeling of Multi-legged Robots." He has completed about 20 projects. He has guided 22 Ph.D.s. He is in editorial board of 14 International Journals. He has been elected as FIE, SMIEEE, MASME.