Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 18(3): 2527–2561.
DOI: 10.3934/mbe.2021129
Received: 12 November 2020
Accepted: 04 March 2021
Published: 15 March 2021

Research article

Multi-objective grasshopper optimization algorithm based on multi-group and co-evolution

Chao Wang[1,2], Jian Li[1,2], Haidi Rao[1,2], Aiwen Chen[1,2], Jun Jiao[1,2], Nengfeng Zou[1,2] and Lichuan Gu[1,2,*]

[1] Anhui Agricultural University, Hefei 230036, China
[2] Key Laboratory of Agricultural Electronic Commerce of the Ministry of Agriculture, Hefei 230036, China

* **Correspondence:** Email: glc@ahau.edu.cn; Tel: 86055165786188; Fax: 86055165786188.

Abstract: The balance between exploration and exploitation is critical to the performance of a Meta-heuristic optimization method. At different stages, a proper tradeoff between exploration and exploitation can drive the search process towards better performance. This paper develops a multi-objective grasshopper optimization algorithm (MOGOA) with a new proposed framework called the Multi-group and Co-evolution Framework which can archive a fine balance between exploration and exploitation. For the purpose, a grouping mechanism and a co-evolution mechanism are designed and integrated into the framework for ameliorating the convergence and the diversity of multi-objective optimization solutions and keeping the exploration and exploitation of swarm intelligence algorithm in balance. The grouping mechanism is employed to improve the diversity of search agents for increasing coverage of search space. The co-evolution mechanism is used to improve the convergence to the true Pareto optimal front by the interaction of search agents. Quantitative and qualitative outcomes prove that the framework prominently ameliorate the convergence accuracy and convergence speed of MOGOA. The performance of the presented algorithm has been benchmarked by several standard test functions, such as CEC2009, ZDT and DTLZ. The diversity and convergence of the obtained multi-objective optimization solutions are quantitatively and qualitatively compared with the original MOGOA by using two performance indicators (GD and IGD). The results on test suits show that the diversity and convergence of the obtained solutions are significantly improved. On several test functions, some statistical indicators are more than doubled. The validity of the results has been verified by the Wilcoxon rank-sum test.

Keywords: multi-objective optimization; meta-heuristics; swam intelligence algorithm; grasshopper optimization algorithm

1. Introduction

Many complex problems in real life are composed of conflicting and influential objectives, they need to be optimized simultaneously as much as possible in given constraints, that is, multi-objective optimization. These problems are very complex, difficult, but also very important. Multi-objective optimization is common and significant in actual life, and it is widely used in production and engineering design [1], job scheduling [2], management and decision-making [3], etc. In the multi-objective optimization problem, each objective restricts each other, it is impossible to have a solution that can make all the objectives achieve the optimal performance. Therefore, for the multi-objective optimization problem, the optimal solution is usually a set of non-inferior solutions, namely Pareto optimal solution set [4]. Therefore, the main tasks of multi-objective optimization are as follows: 1) Finding a set of solutions as close as possible to the Pareto front; 2) Finding a set of solutions as different as possible [4,5].

Meta-heuristic methods have powerful global search capabilities, which design iterative equations by simulating the behavioral characteristics of biological groups or the development and structural characteristics of physical things. Therefore, they are more appropriate for settling complex multi-objective optimization problems [6]. At present, a large number of Meta-heuristic methods have been proposed and improved to solve multi-objective optimization problems, such as, monarch butterfly optimization (MBO) which simplifies and simulates the migration process of monarch butterfly [7], slime mould algorithm (SMA) which is proposed based on the oscillation mode of slime mould in nature [8], Moth search algorithm (MSA) [9] which is a new kind of metaheuristic algorithm inspired by the phototaxis and Lévy flights of the moths, Harris hawks optimization (HHO) which simulates the predatory behavior of Harris hawk [10].

Meta-heuristic methods include two contradictory processes: exploration and exploitation, which need to establish a balance between them [11,12]. When exploitation is enhanced, randomization of the search is increased, which is helpful to avert falling into the local optimal solution in the process of optimization and improve the convergence accuracy. Conversely, when exploration is enhanced, local search capability of the meta-heuristic algorithm is more powerful, and the convergence speed is faster, but the approach is easier to fall into local optimum, especially the diversity of the obtained solutions becomes worse [13,14]. Meta-heuristic methods are population-based search and optimization methods whose efficacy mainly depends on a fine balance between exploration and exploitation [15]. Therefore, the balance between exploration and exploitation has been widely studied in meta-heuristic optimization algorithms [15–20].

The meta-heuristic optimization algorithms should be equipped with special control parameters to adjust exploration and exploitation in different stages of optimization. The control parameters should be designed and adjusted according to the specific optimization problems. Generally, there are two ways to design the control parameters. One is the constant control strategy [21], that is, to keep the control parameters unchanged in the whole iterative search process. Another way is to dynamically adjust the value of control parameters in the iterative search process to achieve the balance between exploration and exploitation [22]. The first way is simple and easy to implement, but it is very difficult to clearly identify the exploration and exploitation phases. It is generally considered that the strategy of more exploration in the early phase and more exploitation in the later phase is usually employed in the whole search process，such as linear control strategy [1,14], and nonlinear control strategy [22,23]. Therefore, the method of dynamically adjusting the value of control parameters is widely employed in meta-heuristic optimization algorithms. As discussed in [24], a proper balance between exploration and exploitation may drive the search process towards better performance. The balance between exploration and exploitation is critical to the performance of a meta-heuristic optimization method. However, a scheme of adjusting control parameters that performs better in problem A might not work effectively in problem B. How to balance the relationship between exploration and exploitation in complex optimization problems has become the goal of meta-heuristic algorithms design or improvement, which is still an open question. For this problem, the solution in this paper is that different values of the same parameters can be employed to different subpopulations of a meta-heuristic optimization method at the same time to ensure that a suitable value of parameters can be employed for the specific optimization problem.

As one of the dominant modern meta-heuristic optimization algorithms, grasshopper optimization algorithm (GOA) has been successfully applied to various optimization problems in several fields. The good performance of GOA has been proved in [1]. It has one control parameter, which balances between exploration and exploitation. Therefore, its advantages are over the other optimization algorithms, including ease of implementation, speed in searching, and ease of modifying algorithm components [25]. However, it suffers from slow convergence and is prone to getting stuck in local optima [25,26]. To resolve the issues above, a multi-group and co-evolution framework was proposed to improve GOA and apply to multi-objective optimization problems, which draws on the idea of co-evolution strategy in [24]. In order to achieve population diversity and evolutionary adaptability, each subpopulation of GOA is assigned a different parameter c, including linear adaptation, cosine adaption, arc adaption, etc. At the same time, the subpopulation is dynamically updated in the optimization process.

The main contributions of this research are as follows:

1) A multi-group and co-evolution framework is proposed to archive a fine balance between exploration and exploitation of swarm intelligence algorithm

2) Multi-objective grasshopper optimization algorithm base on the multi-group and co-evolution framework is developed to improve the convergence and diversity of optimal solutions.

3) Detailed experiments are designed on several benchmark functions to demonstrate the effectiveness of the proposed methods.

The rest of this paper is organized as follows. Section 2 introduces the related work about multi-objective optimization problems. Section 3 describes the definition of the multi-objective optimization problem to be solved in this paper. In Section 4, the multi-group and co-evolution framework, and the multi-objective grasshopper optimization algorithm based on this framework are illustrated in detail. Section 5 includes presentation and analysis of outcomes on test functions. Finally, Section 6 concludes the main work of this paper and suggests the next step of research work.

2. Related works

Multi-objective optimization problems do not have a globally unique optimal solution, but to discover a cluster of optimal solutions. Therefore, the key to solving multi-objective optimization problems is to find a set of non-inferior solutions in the solution space, that is, the Pareto front. The approaches for settling multi-objective optimization problems can be divided into 5 categories [5]: decomposition methods [27,28], interactive methods [29,30], fuzzy methods [31,32], decision aid methods [33,34], and methods using meta-heuristics [35,36].

The meta-heuristic is an iterative generation progress that combines different heuristic algorithms to explore the search space. These algorithms are applied to settle complex optimization problems where traditional heuristics and optimization methods are not capable of being effective and efficient [37]. Constructing a dynamic balance mechanism between diversification and intensification is a key step. Diversification can explore the broader search space, while intensification is an in-depth exploitation of specific areas by using the accumulated experience in the previous search process. Meta-heuristic methods can quickly explore the areas containing high-quality solutions in the search space, meanwhile, they do not waste time to exploit areas that have been exploit before or can not find high-quality solutions. There are three meta-heuristic strategies: a dominance-based approach, an indicator-based alternative, and an adaptive proposal that incorporates both multi-objective strategies (dynamically allocating more resources to the most successful strategy during the execution) [38]. Meta-heuristic methods have widespread applications. The Non-dominated Sorting Genetic Algorithm (NSGA-II) is applied to settle the multi-objective optimization problem of telecommunication network, and then use the Choquet integral measure based on utility function to make the final choice [39]. A meta-heuristic method for RNA inverse folding problem is designed by Tian et al. [40], in which RNA inverse folding problem is defined as a multi-objective optimization problem, and the similarity between the target structure and the predicted structure is employed as a constraint. In [41], a novel flexible job shop scheduling problem based on linear programming model is built to schedule the spool fabrication activities. Then priority dispatching rules based heuristic scheduling approach is utilized to settle the problem. In [42], an extended Multi-row facility layout problem (MRFLP) has been studied, and the genetic algorithm is applied to resolve the optimization problem.

Recently, many meta-heuristic nature-inspired algorithms are proposed. They are nature-inspired and population-based optimization algorithms, which have been developing and adapting in varying ways, such as learner performance based behavior algorithm (LPB) [43], Dragonfly Algorithm (DA) [44], cat swarm optimization (CSO) [45], Backtracking search optimization algorithm (BSA) [46], Donkey and Smuggler Optimization Algorithm (DSO) [47], Fitness Dependent Optimizer (FDO) [48] and IFDO [49], Particle Swarm Optimization Algorithm(PSO) [50], Firefly Algorithm (FA) [51], Grasshopper Optimization Algorithm (GOA) [1], and several hybrid algorithm of different nature-inspired algorithms including A hybrid of Shuffled Frog Leaping Algorithm and Genetic Algorithm (SFGA) [52], WOAGWO [53] hybridizing Whale optimization algorithm (WOA) [54] with Grey wolf optimization (GWO) [55], and a many-objective multi-agent coordination optimization algorithm (MOMCO) [56] hybridizing with EA.

Swarm intelligence is an important meta-heuristic optimization method, which has been widely used in different fields as a very important optimization technology. Swarm intelligence optimization algorithm is a combination of randomness and some rules to solve the optimization problem, through the simulation of natural phenomena. This kind of algorithm has a general optimization process: firstly, the population is initialized randomly, and then the search direction of every individual in the population is guided according to the rules. Then when termination conditions are met (such as, the maximum number of iterations is reached), the algorithm stops, and the final solution is the global optimal solution. Therefore, the composition of their computational complexity is similar, which is composed of three main parts: population initialization, population individual updating and fitness calculation. Generally, it is closely related to the number of iterations *M*, the size of the population *N*, the dimension of the decision space *D*, the complexity of updating function $f_1(x)$ of the individual and the complexity of the fitness function $f_2(x)$. Therefore, the computational complexity of swarm intelligence can be presented as $O(N) + O(MND) \times O(f_1(x)) + O(MN) \times O(f_2(x))$, where the big oh notation is used to show the computation complexity. $O(N)$ is computational complexity of population initializing, $O(MND) \times O(f_1(x))$ is the computational complexity of population updating, and $O(MN) \times O(f_2(x))$ is the computational complexity of fitness calculation. Such as the computational complexity of PSO [50] is $O(N) + O(MND) \times O(N) + O(MN) \times O(N)$), which takes $O(MN^2D)$; the computational complexity of WOA [54] and GOA [1] is also takes $O(MN^2D)$ respectively.

Table 1. Literature review of adjustment methods of parameter c in GOA.

Reference	Control strategy	Method
[1,14,57,58,60–63]	LCS	$c = c_{max} - m\frac{c_{max} - c_{min}}{M}$
[64]	NCS	$c = \left(cos\left(\frac{\pi m}{M}\right) + c_{max}\right)\left(\frac{c_{max} + c_{max}}{2}\right)$
[64]	NCS	$c = \left(\frac{c_{max} - m}{M}\right)^2$
[59]	NCS	$x_{i+1}^d = c_1(t)\left\{\sum_{\substack{j=1 \\ j\neq i}}^{N} c_2(t)\frac{ub_d - lb_d}{2} s(\lvert x_j^d - x_i^d \rvert)\frac{x_j - x_i}{d_{ij}}\right\} + T_d$ where $c_1(t)$ and $c_2(t)$ are the values generated from a wide variety of different chaotic maps in the *t-th* iteration.
[65]	NCS	$c = c_{min} + (1 - c_{min})(1 - m^w)^{1/w}$ where the values of w vary between 0 and 1, if w is greater than 0.5, exploration is more important, but if w is less than 0.5, exploitation is more important [65].
[66]	NCS	$c = \begin{cases} \delta[1 + k_1(f_g/(f_{max} - f_{min} + f_{avg}))], & f_g \geq f_{avg} \\ k_2, & f_g \leq f_{avg} \end{cases}$ where, f_{max} and f_{min} determine the maximum/minimum value of all agents fitness when AGOA do a search operation, f_{avg} determines the average fitness, f_g denotes the average fitness of the three parents in selection operation. δ, k_1 and k_2 notify the constant value in between 0 to 1[66].

∗ Formal notations used in this paper: c is a decreasing coefficient to shrink the comfort zone, repulsion zone, and attraction zone, which maintains the balance between exploration and exploitation of GOA; m and M indicate the current iteration and total number of iterations respectively; c_{max} and c_{min} represent the maximum and minimum values of parameter c; N is the number of grasshoppers; ub_d and lb_d are the upper and lower boundary of the D-dimensional search space separately; x_i represents the position of the *i-th* grasshopper.

Among these swarm intelligence algorithms, GOA has been proved to have good performance in literature [1]. Therefore, there is a lot of literature on GOA based improvements and related applications. A comprehensive survey of the GOA is summarized in [25], which analyzes several modified versions of GOA and its applications in different fields in detail. For the purpose of ameliorating the optimization performance, GOA_jDE [57] which combines GOA and jDE [58] is proposed. GOA_jDE can greatly improve the convergence efficiency, convergence speed and calculation precision on the benchmark test in [57]. The chaos theory is also introduced into the optimization process of GOA, which employs chaotic maps to keep the exploration and exploitation progress balance of GOA and accelerate its global convergence speed [59]. A multi-objective grasshopper optimization algorithm for robot path planning in static environments was proposed to optimize several indexes such as cost, distance, energy or time [60].

In these GOA based optimization methods, the control parameter c is a significant parameter to maintain a balance between exploration and exploitation. The adjustment of parameter c can be divided into linear control strategy (LCS) and nonlinear control strategy (NCS), according to the classification method in [22]. In the original GOA, the linear control strategy is adopted for the control parameter c, which linearly decreases from the maximum value to the minimum value with the increase of the number of iterations. LSC transits linearly from the exploration phase to the exploitation phase. However, in order to balance exploration and exploitation more reasonably, some nonlinear control strategies (NCS) are proposed, such as cosine adaption, arc adaption, etc. A comprehensive table of adjustment methods of parameter c in GOA is presented in Table 1.

In the optimization process of meta-heuristic methods, how to find a fine balance between exploration and development is critical. Each specific balance control strategy has a good performance in a certain kind of optimization problems, but how to select the appropriate balance control strategy for specific complex optimization problem is very difficult, and it needs a very deep understanding of the optimization problem. In order to solve the above issues, a framework is designed in this paper, which integrates a variety of balance control strategies to achieve adaptive selection of the appropriate balance and improve the optimization effect.

3. Multi-objective optimization problem description

The ordinary form of multi-objective optimization problem definition is as follows [4]:

$$Minmize: F(x) = \{f_1(x), f_2(x), \dots, f_M(x)\} \tag{1}$$

where $x = [\ x_1, x_2, \dots, x_D]$ is a point in the D-dimensional decision space (search space), $f = [f_1, f_2, \dots, f_M]$ is an objective space with M optimization objectives.

The solutions of multi-objective optimization problems are usually a group of non-inferior solutions called Pareto optimal solutions. The Pareto front is the boundary defined by the set of all point mapped from Pareto optimal solutions. Equations (2) and (3) define the set of Pareto optimal solutions P_s and Pareto front P_f separately [4]:

$$P_s = \left\{x^*: \left[(\forall i \in \{1,2,\dots,k\}, x: f_i(x) \le f_i(x^*)) \bigwedge (\exists j \in \{1,2,\dots,k\}: f_j(x) < f_i(x^*))\right]\right\} \tag{2}$$

$$P_f = \{f(x^*): x^* \in P_s\} \tag{3}$$

In multi-objective optimization problems, the sub-objective functions may be uncorrelated or conflicting. It is not possible to have a certain solution that can make all the sub-objectives achieve optimal performance. Therefore, the primary task of all multi-objective optimization algorithms is to seek out as many Pareto optimal solutions as possible [5]. The proposed method of this paper has two objects: 1) finding a set of solutions $f(x')$ which converge to the Pareto front P_f as close as possible, as shown in Eq (4); 2) finding a group of solutions as diverse as possible, as shown in Eq (5).

$$\left|f(x' - P_f)\right|_t < \varepsilon \tag{4}$$

$$Difference(f(x')) > \sigma \tag{5}$$

In this paper, the multi-group and co-evolution framework of swarm intelligence algorithm is designed to further achieve these two objects. By grouping the entire population into different subpopulations, the proposed framework increases the diversity of population and the randomness of search, to meet the requirements of Eq (5). In the iterative process, the co-evolution mechanism between subpopulations is established to ameliorate the convergence speed and accuracy of swarm intelligence algorithm, satisfying the requirements of Eq (4) as much as possible. The solution space of multi-objective optimization problem is transformed into the search space of the swarm intelligence algorithm, and the metrics of x' is also transformed into the search conditions and evolution criterions of the swarm intelligence algorithm. So, in the multi-objective optimization problems, the convergence speed refers to the number of iterations t needed to find several solutions satisfying Eq (4). The convergence accuracy is determined by ε in Eq (4), the smaller ε is, the higher the convergence accuracy is. The diversity of Pareto optimal solutions is determined by Eq (5). Each solution should keep a certain difference or distance. In the experiment, this paper does not directly and independently determine the parameters ε and σ, but indirectly chooses the multi-objective optimal solutions according to the overall performance of their convergence and diversity on the real Pareto front.

4. MOGOA based on the multi-group and co-evolution framework

Firstly, the multi-group and co-evolution framework of swarm intelligence algorithm is presented in Section 4.1, including the design of the grouping strategy of the population, the co-evolution strategy among subpopulations, and the selection strategy of key parameters. In Section 4.2, we first briefly introduce GOA, and then illustrate how to integrate GOA into the framework in detail. MOGOA based on the multi-group and co-evolution framework is described in Section 4.3.

4.1. The multi-group and co-evolution framework

The multi-group and co-evolution framework is an optimization mechanism of swarm intelligence algorithm. It can keep the exploration and exploitation balance in the search process and ameliorate the speed and accuracy of convergence by building variety of subpopulations and establishing information interaction mode between subpopulations. As shown in Figure 1, the multi-group and co-evolution framework of a swarm intelligence algorithm is divided into two key components: the grouping mechanism and the co-evolution mechanism. The grouping mechanism includes two steps: population division and parameter setting. The co-evolution mechanism includes two steps also: communication and feedback between subpopulations.

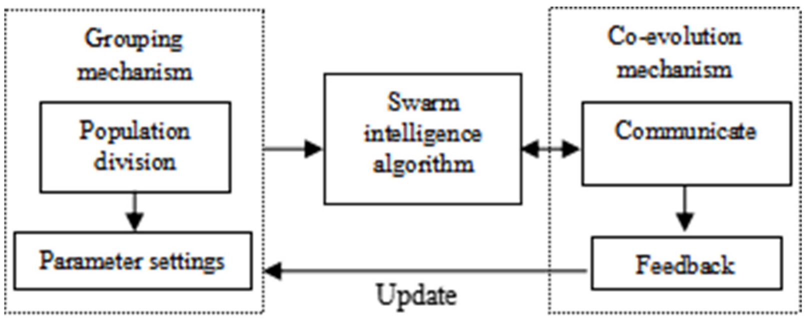

Figure 1. Multi-group co-evolution framework of swarm intelligence algorithm.

4.1.1. Grouping mechanism

When the optimization objective is determined, the swarm intelligence algorithm will select and determine the population size, control variables and other key parameters of the population to search the optimization space. Depending on the parameter change mechanism in the search process, the exploration and exploitation of the swarm intelligence algorithm are constantly compromised. A commonly used method is: in the incipient stage of search, the exploration is more important which make the search agent cover more search space; and then the exploitation is gradually emphasized with the iterative process of the algorithm, the optimal solution is searched in the local space [4]. After the population is determined, the search agents start to work according to the established search strategies in the whole search iteration process. Therefore, all search agents in the population and their search strategies lack difference. For the purpose of increasing the difference and diversity of search agents, this paper proposes a grouping mechanism, which divides the entire population into different subpopulations. Each subpopulation can be set different key parameters and search strategies.

The entire population can be divided into different subpopulations in different ways. According to the size of subpopulation, there are average divisions, random divisions and so on; the setting of the initial parameters and search strategy of each subpopulation can also be achieved by random or fixed assignment. Based on GOA, this paper focuses on the research and implementation of two grouping mechanisms: fixed and random assignment of parameters under the condition of average population division.

4.1.2. Co-evolution mechanism

Different subpopulations have different optimization paths, and different search agents also produce different search information in each subpopulation. The proposed co-evolution mechanism is used to determine the interaction pattern of this information, including those within and between subpopulations. After different subpopulations have completed the search in parallel, they can share the information of the optimal solution and the worst solution found in the current iteration, and constantly adjust the search strategy. Meanwhile, the performance of different subpopulations is feed back to the grouping mechanism, and then the size and some key parameters of subpopulations are updated according to the feed information to further improve the efficiency of the swarm intelligence algorithm.

4.2. GOA based on the multi-group and co-evolution framework

The idea of grasshopper optimization algorithm is briefly described firstly, then the implementation of GOA with the grouping and co-evolution mechanism is described in detail, and the pseudo code of GOA based on the multi-group and co-evolution framework is also given.

4.2.1. Grasshopper optimization algorithm

Following is presenting the mathematical model employed to simulate the swarming behavior of grasshoppers [1]:

$$X_i = r_1 * S_i + r_2 * G_i + r_3 * A_i \tag{6}$$

where X_i represents the position of the *i-th* grasshopper, S_i signifies social interaction, G_i defines the gravity force on the *i-th* grasshopper, A_i shows the wind advection. random numbers r_1, r_2 and r_3 selected from [0,1] are used to provide random behaviors.

S_i is an important component of the model [1], as follows:

$$S_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} s(|x_j - x_i|) \frac{x_j - x_i}{d_{ij}} \tag{7}$$

where d_{ij} represents the distance between *i-th* and *j-th* grasshopper, *N* is the number of grasshoppers.

In Eq (7), the function $s(r)$ signifies the social forces [1], as follows:

$$s(r) = f e^{\frac{-r}{l}} - e^{-r} \tag{8}$$

where *f* defines the intensity of attraction and *l* indicates the attractive length scale.

G_i is calculated as follows [1]:

$$G_i = -g * e_g \tag{9}$$

where *g* is a constant which is used to adjust the effect of gravity and e_g is a unity vector towards the center of earth.

A_i is calculated as follows [1]:

$$A_i = u * e_w \tag{10}$$

where *u* is a constant which is used to adjust the effect of drift and e_w is a unity vector determining the direction of wind.

So, Eq (6) can be rewritten as follows [1]:

$$x_i^d = \sum_{\substack{j=1 \\ j \neq i}}^{N} s(|x_j^d - x_i^d|) \frac{x_j - x_i}{d_{ij}} - g * e_g + u * e_w \tag{11}$$

To solve optimization problems, G_i is omitted and A_i is replaced by optimization target as the wind direction in Grasshopper Optimization Algorithm (GOA) [1]. The above model is adjusted as follows:

$$x_{i+1}^{d} = c\left\{\sum_{\substack{j=1\\ j\neq i}}^{N} c\frac{ub_d - lb_d}{2} s(|x_j^d - x_i^d|)\frac{x_j - x_i}{d_{ij}}\right\} + T_d \tag{12}$$

where ub_d and lb_d are the upper and lower boundary of the D-dimensional search space separately. T_d is the optimal solution of the objective function in the current iteration as the wind direction. c is a key parameter in GOA, which is proportional to the amount of iterations. The parameter c helps to balance the value of repulsion/gravitation between grasshoppers dynamically [1].

If the parameter c decreases too fast in the inchoate stage of GOA, it will make an insufficient exploration. Conversely, if the parameter c decreases too slowly in the later stage of GOA, the exploitation of GOA will be insufficient. For the purpose of keeping the exploration and exploitation balance of GOA, this paper ingrates GOA into the multi-group and co-evolution framework to effectively balance local and global search ability of GOA.

4.2.2. Grouping algorithm

Grouping algorithm for GOA includes two key parameters setting: subpopulation number and the parameter c of GOA.

• The initial subpopulation number and size

There are several ways to divide population of GOA into different groups, such as average division, dynamic division and random division. This paper adopts the average division method, which facilitates the comparison of the final results during the iterative update process, such as Eq (13). The number and the size of subpopulation are related to the specific optimization problem.

$$S = (S_1,\ S_2, \dots, S_{ns}) \tag{13}$$

where the whole population *S* are divided into *ns* subpopulations, each of which have the same size.

• Settings of the parameter c of GOA

In this paper, three forms are selected to set the parameter c of GOA [10,57], as follows:

$$c_1 = c_{max} - m\frac{c_{max} - c_{min}}{M} \tag{14}$$

$$c_2 = \left(cos\left(\frac{\pi m}{M}\right) + c_{max}\right)\left(\frac{c_{max} + c_{max}}{2}\right) \tag{15}$$

$$c_3 = \left(\frac{c_{max} - m}{M}\right)^2 \tag{16}$$

where c_1, c_2 and c_3 are three different forms of parameter c, c_1 is a linear adaptation form of LCS, c_2 is a cosine adaption form of NCS, c_3 is an arc adaption form of NCS; c_{max} means the maximum, c_{min} means the minimum, *m* is the current iteration number, *M* is the maximum iteration number, in this paper, c_{max} and c_{min} are set to 1 and 0.00001, respectively.

We use a fixed assignment strategy and an equal probability random assignment strategy to allocate different parameter c to each subpopulation in this paper.

The fixed assignment strategy adopts the following methods:

$$c_{S_i} = c_j, with\ \ c_j \in (c_1, c_2, \dots, c_n) \tag{17}$$

where S_i presents the *i-th* subpopulation, c_{Si} is the form of parameter c of the *i-th* subpopulation, n is the number of parameters c, c_j is one of n different forms of parameter c.

Before the search process starting, each subpopulation S_i fixedly chooses a corresponding parameter c_j which remains unchanged in the whole optimization process.

The random assignment strategy adopts the following methods:

$$c_{S_i} = c_r, with\ \ r = rand(1, n), c_r \in (c_1, c_2, \dots, c_n) \tag{18}$$

where r is a random integer between 1 and n, n is the number of parameters c that can be chosen.

In each iteration, every subpopulation can choose a different form of parameter c with equal probability according to the calculated random number in Eq (18). The Grouping algorithm not only enhances the diversity of GOA's population, but also avoids GOA from falling into a local optimum. The multiplicity of population also increases the adaptability of the algorithm to solve optimization problems with different characteristics.

Algorithm1. GOA-MC

```
——————Initialization operation——————
1.Initialize the grasshopper populationS
——————Grouping operation——————
2. Divide the entire population into ns subpopulations $S = (S_1,\ S_2, \dots, S_{ns})$
3.Calculatethe initial fitness of each subpopulations
4.[TargetFitness, TargetPosition] = Select the initial optimal solution and the corresponding optimal position in all subpopulations
——————Co-evolutionary operation——————
5. while (m< Max number of iterations)
6.      update the evolution direction of entire population by TargetPosition
7.      for each subpopulation $S_i$
8.          Select parameter  c using Eq. (16) for subpopulation $S_i$
9.          Update the position of current grasshopper by the Eq. (12)
10.         Calculate the current fitness based on object function and current grasshopper position
11.     end for
12.     Update [TargetFitness, TargetPosition] if there is a better solution in entire population
13.     m=m+1
14. end while
15. Return :  The final optimal solution [TargetFitness, TargetPosition]
```

Figure 2. The pseudo code of GOA-MC.

4.2.3. Co-evolutionary algorithm

The co-evolutionary algorithm builds connections between the different subpopulations, realizes the information interaction between subpopulations, and then adjusts the evolution direction of GOA. Before running the next iteration, we compare the advantages and disadvantages of the optimal solution obtained from each subpopulation in the current iteration, in order that update the optimal solution of the entire population. The optimal solution is assigned to each subpopulation as the new evolution direction of GOA. Therefore, after each iteration, all subpopulations determine a new evolution direction again based on the results of the previous information interaction. In the frequent iterative process, subpopulations constantly exchange information to achieve the purpose of co-evolution. The co-evolutionary algorithm increases the probability that the population converges to the optimal solution. The optimal solution obtained is more intensification in each independent running.

4.2.4. GOA based on multi-group and co-evolution

The pseudo code of GOA based on the multi-group and co-evolution framework (GOA-MC) is shown in Figure 2. According to grouping algorithm, the group operation first divides the entire population into ns subpopulations. In the co-evolutionary operation, the evolution direction of entire population is updated according to the optimal solutions of all subpopulations in each iteration. Then using Eq (12) update the position of all grasshoppers according to the direction of evolution. In each iteration, the parameter c of subpopulations can change using Eq (18). The update of positions of grasshoppers runs until the termination condition is met. Finally, the optimal fitness and position of the objective function are given.

4.3. MOGOA based on the multi-group and co-evolution framework

Results of the experiment in Section 5.1 verify that there is a remarkable enhancement in convergence speed and accuracy of GOA-MC algorithm compared to the initial GOA algorithm. Therefore, a MOGOA-MC method is proposed which integrate MOGOA [29] into the multi-group and co-evolution framework, as shown in Figure 3. Using the same structure as MOGOA, this paper also constructs an external archive to keep optimal solutions from all subpopulations in each iteration and chooses the first *n* best performing optimal solutions as the final Pareto solution set in lines 5–6. Using the idea of co-evolutionary operation in algorithm 1, the evolutionary direction of all subpopulations is updated in line 7.

The final results of the multi-objective optimization problem are a group of solutions, which can not be directly compared with each other to get one optimal solution. So, we construct an external archive to store the optimal solution set, then compare and update the archives of all subpopulations together to obtain a global optimal solution set by adopting the elitist scheme in line 6. When the total amount of optimization solutions searched is greater than the size defined by the external archive, the average distance between each solution and all other solutions is figured using Eq (19), and the distances are sorted. The top n solutions with the largest average distance are selected and kept in the archive. Each new solution obtained from the next iteration needs to be compared with the average distance of other solutions in the archive, so as to constantly update the archive.

$$\mathrm{A\_dist}(\mathrm{x_i}) = \frac{1}{\mathrm{K}}\sum_{\substack{\mathrm{j}=1 \\ \mathrm{j}\neq\mathrm{i}}}^{\mathrm{K}} \mathrm{dist}(\mathrm{x_i} - \mathrm{x_j}) \tag{19}$$

where $A\_dist(x_i)$ refers to the average distance between optimal solution x_i and all other solutions, K is the size of the external archive.

This method of selecting the optimal solutions has a disadvantage: in the process of archive updating, it can not guarantee that the final selected optimization solutions are those with the largest average distance among all searched optimal solutions. In other words, among all the optimal solutions obtained, the solutions with larger average distance may be deleted in the update process. This disadvantage can be improved by increasing the size of external archive, but the increase of capacity will increase the cost of optimal solutions ranking. The size of external archive is related to the specific optimization objectives, and a balance needs to be made between the cost of computation and the diversity of optimal solutions.

Algorithm 2. MOGOA-MC

```
1.Initialize and Divide the grasshopper populationS  by initialization and grouping operation of Algorithm 1 (line1-4)
2.  Initialize the external archive
3.while (m< Max number of iterations)
4.      Calculate fitness based on current grasshopper position
5.      Update the external archive using the optimal solutions obtained from all subpopulations
6.      Compare he optimal solutions in the archive and keep the top n best using Eq. (17)
7.      Update the evolutionary direction of all subpopulations by Co-evolutionary operation of Algorithm 1 (line7-11)
8.      m=m+1
9.  end while
10. Return :  Pareto solution set
```

Figure 3. The pseudo code of MOGOA-MC.

5. Results on test functions

This section first benchmark the performance of the proposed GOA-MC algorithm to verify the effectiveness of the multi-group and co-evolution framework in Section 5.1. Then in Section 5.2, several standard multi-objects test functions with different characteristics are applied to verify the performance of MOGOA-MC algorithm. The details of test functions employed in this work are presented in Tables A1–A6 of the Appendix. For the purpose of verifying the outcomes, GOA [1] and MOGOA [14] which have very good performance in the literatures of optimization problems are employed to compare with GOA-MC and MOGOA-MC respectively. All the experiments were carried out in this PC Configuration: System, Windows 10; CPU, 3.00 GHz; RAM, 16.00 GB; Language, Matlab 2016.

5.1. Quantitative results and discussion of GOA-MC

For the purpose of verifying the improvement of convergence accuracy, convergence speed and search ability by the multi-group and co-evolution framework, a series of test functions [1,67] with different characteristics are applied, where test functions F1–F7 are unimodal benchmark functions, F9–F13 are multimodal benchmark functions, and F14–F19 are composite benchmark functions. Then the sensitivity analysis of the main parameters is carried out to verify the effectiveness of this framework.

5.1.1. Discussion of computational results

We compared and analyzed the performance of GOA with three different forms of parameter c and the GOA-MC with two different strategies. GOA-1, GOA-2, and GOA-3 are the original grasshopper optimization algorithm without using the multi-group and co-evolution framework. GOA-1, GOA-2, and GOA-3 respectively use the linear self-adaptive, cosine self-adaptive, and arc self-adaptive parameter c in Eqs (14)–(16). GOA-F and GOA-R are based on the multi-group and co-evolution framework respectively using fixed and randomly assigned parameter c in Eqs (14)–(16). For the sake of fairness, the main control parameters used in these algorithms refer to the parameter settings in [1] and [14], which are displayed in Table 2. Except for the number of subpopulations, other parameter settings are the same.

Table 2. Initial values for the control parameters of algorithms.

Algorithm	Parameter	Value
GOA-1	size of population	120
GOA-2	number of subpopulations	1
GOA-3	number of iterations	300
	c_{max}, c_{min}	1, 0.00001
GOA-F	size of population	120
GOA-R	number of subpopulations	3
	number of iterations	300
	c_{max}, c_{min}	1, 0.00001

Each test function runs 20 times independently for generating the statistical results in Table 3 and Figure 4. The average convergence curve of F1–F19 are presented in Figure 4. The steeper the curve descends, the faster the convergence rate will be. The closer the curve is to the x-axis, the better the convergence accuracy is. Compared with GOA-1, GOA-2 and GOA-3, GOA-F and GOA-R have significant performance improvement for most benchmark test functions. The way regulating the balance mechanism of exploration and exploitation is different by different forms of parameter *c*, such as LCS or NCS. In the GOA-F and GOA-R, subpopulations with different forms of parameter *c* search the optimal value at the same time. Therefore, in the process of searching the optimal value, a variety of balance mechanisms are used, then the search information exchange among subpopulations is completed through the co-evolution mechanism, which improves the ability of GOA to find the optimal value quickly and accurately. These results in Table 3 also prove that the optimization capability of GOA can be effectively improved by the multi-group and co-evolution framework.

For unimodal benchmark test functions F1–F7 which have only one global optimum, the outcomes in Table 3 and Figure 4 prove that the multi-group and co-evolution framework significantly improves convergence accuracy and speed of GOA. On the multimodal benchmark functions F8–F13 with several local optimal solutions, except for F9, GOA-F and GOA-R outperform others. The results of F8–F13 in Figure 4 also show the superiority of the multi-group and co-evolution framework in convergence speed. Due to the existence of local optimal solutions, the optimization algorithm needs to further balance exploration and exploitation and jumps out of local optimum through exploration. Therefore, the comprehensive performance in the multimodal benchmark functions shows that the grouping mechanism of the framework can further improve the exploration capacity of the algorithm, and the co-evolution mechanism simultaneously ensures the exploitation ability.

Table 3. Results of benchmark test functions.

	GOA-1		GOA-2		GOA-3		GOA-F		GOA-R	
	AV.	STD.	AV.	STD.	AV.	STD.	AV.	STD.	AV.	STD.
F1	2.1065 × 10^{-8}	6.8522 × 10^{-16}	2.2761 × 10^{-13}	9.6419 × 10^{-26}	4.2179 × 10^{-15}	4.8483 × 10^{-29}	7.3512 × 10^{-15}	2.1538 × 10^{-28}	**4.5877 × 10^{-16}**	**5.3536 × 10^{-31}**
F2	0.3737	0.2482	0.5282	0.8587	0.7809	1.1635	**0.0116**	**6.4328 × 10^{-4}**	0.03344	0.0032
F3	2.1711 × 10^{-7}	8.7347 × 10^{-14}	9.3245 × 10^{-12}	2.7784 × 10^{-22}	2.3598 × 10^{-14}	6.8114 × 10^{-28}	3.0693 × 10^{-11}	7.9237 × 10^{-22}	**1.8549 × 10^{-14}**	**2.1420 × 10^{-28}**
F4	8.5561 × 10^{-5}	1.2551 × 10^{-9}	1.2373 × 10^{-7}	1.2896 × 10^{-15}	3.5723× 10^{-8}	6.3449 × 10^{-16}	1.2469 × 10^{-6}	9.0957 × 10^{-13}	**2.583 × 10^{-8}**	**2.0412 × 10^{-16}**
F5	7.6481	135.3689	30.3799	3.5077 × 10^{3}	2.923	8.5927	0.47497	**0.1125**	**1.9511**	4.1055
F6	2.9434 × 10^{-8}	2.1365 × 10^{-16}	2.3324 × 10^{-14}	2.3266 × 10^{-28}	2.9358 × 10^{-15}	7.4365 × 10^{-30}	5.721 × 10^{-12}	4.7247 × 10^{-23}	**1.2923 × 10^{-15}**	**3.7892 × 10^{-31}**
F7	0.0120	2.4379 × 10^{-4}	0.0083	4.6566 × 10^{-5}	0.0216	0.0014	**0.0003**	**4.0160 × 10^{-8}**	0.0011	8.4289 × 10^{-7}
F8	−1905.8277	1.7892 × 10^{4}	−1663.675	6.1265 × 10^{-4}	−1651.5299	8.2914 × 10^{3}	−1822.201	**4.6223 × 10^{3}**	**−1928.4939**	2.5448 × 10^{4}
F9	7.2976	19.5902	**4.544**	**10.9820**	5.966	31.0130	6.4971	67.4180	5.4234	27.8018
F10	1.1123	0.7640	0.80252	1.0805	0.6585	0.7227	**8.5829 × 10^{-7}**	**6.6770 × 10^{-13}**	0.32926	0.4818
F11	0.18848	0.0069	0.16064	0.0111	0.21093	0.0183	**0.10766**	**0.0035**	0.16875	0.0120
F12	6.4353 × 10^{-5}	2.0701 × 10^{-8}	1.2925 × 10^{-8}	7.9405 × 10^{-16}	6.0977 × 10^{-10}	1.7061 × 10^{-18}	6.8682 × 10^{-9}	2.3364× 10^{-16}	**2.2574 × 10^{-12}**	**1.3613 × 10^{-23}**
F13	0.0044	3.6223 × 10^{-5}	6.9124 × 10^{-10}	2.0558 × 10^{-18}	1.7435 × 10^{-10}	1.5146 × 10^{-19}	3.7631 × 10^{-9}	6.8730 × 10^{-17}	**1.3716 × 10^{-12}**	**8.7490 × 10^{-24}**
F14	0.998	7.0566 × 10^{-31}	0.998	4.9304 × 10^{-32}	0.998	2.4652 × 10^{-32}	0.998	2.4652 × 10^{-32}	0.998	2.4652 × 10^{-32}
F15	0.016592	7.1283 × 10^{-5}	0.0052	7.1981 × 10^{-5}	0.0049	7.4909 × 10^{-5}	**0.0007**	**5.1379 × 10^{-8}**	0.0012	4.2462 × 10^{-7}
F16	−1.0316	1.8413 × 10^{-24}	−1.0316	0	−1.0316	2.4652 × 10^{-32}	−1.0316	1.2326 × 10^{-32}	−1.0316	3.6978 × 10^{-32}
F17	0.3979	1.0239 × 10^{-15}	0.3979	1.9615 × 10^{-24}	0.3979	2.9031 × 10^{-29}	0.3979	6.2699 × 10^{-28}	0.3979	0
F18	3	8.1748 × 10^{-21}	3	4.4275 × 10^{-29}	3	4.4866× 10^{-30}	3	7.6470 × 10^{-29}	3	1.0354 × 10^{-29}
F19	−3.8628	7.7236 × 10^{-6}	−3.8628	5.4281 × 10^{-12}	−3.8628	3.7091 × 10^{-12}	−3.8628	2.0292 × 10^{-15}	−3.8628	5.3004 × 10^{-15}

*Notes: AV.: Average fitness value; STD.: Standard Deviation of fitness values obtained from 20 times independently running

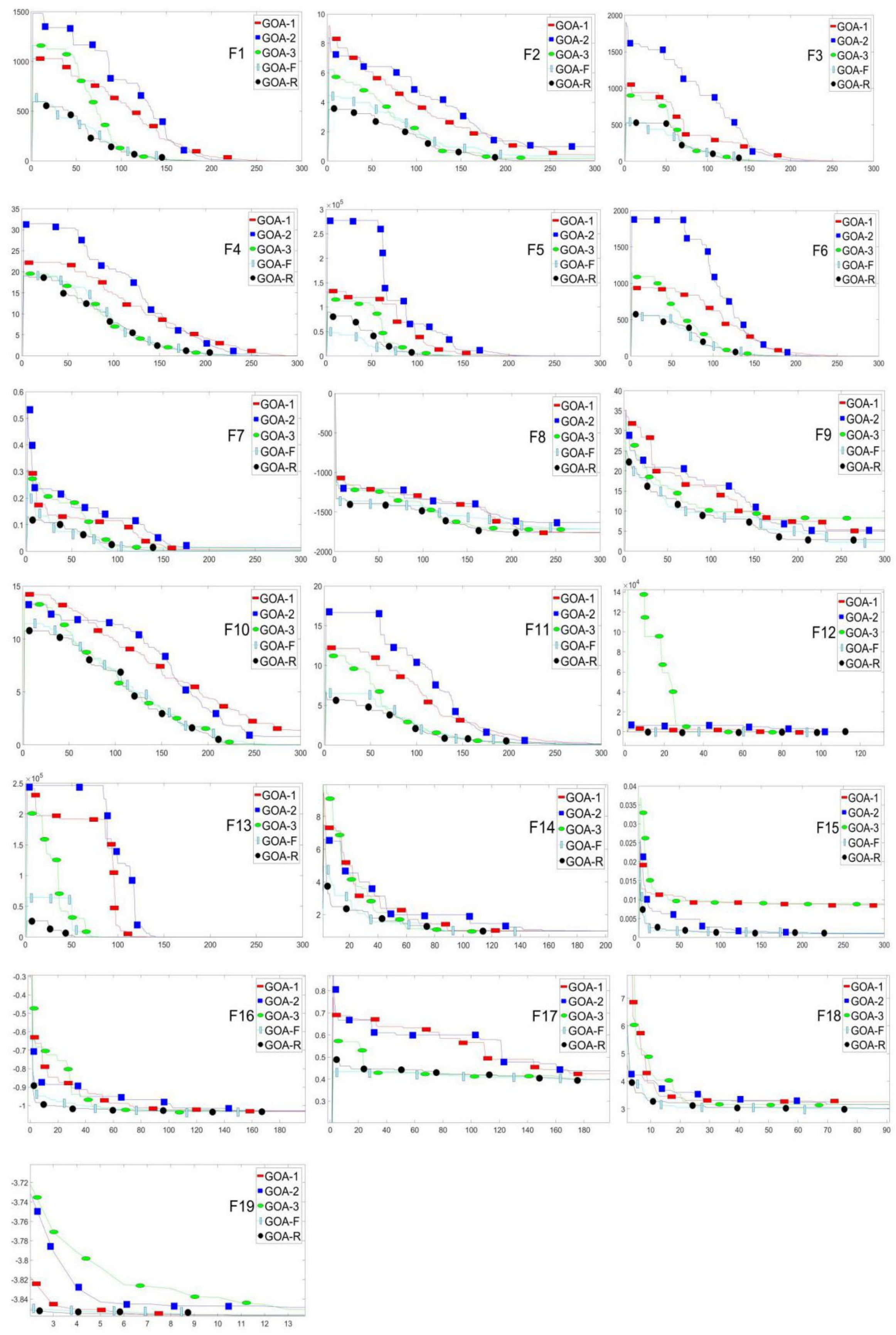

Figure 4. Average convergence curve of F1–F19.

Composite benchmark functions are more complex than other general multimodal benchmark functions. According to the composite benchmark functions F14–F19 in Table 3, it can be observed that all the algorithms have generated the same best results. In the process of calculation, we found an interesting phenomenon: all five algorithms will converge to the same optimal value after a certain amount of iterations, even if we adjust the parameters in Table 2 by a large margin. But in the convergence speed, they are still quite different. As can be observed from Figure 4, GOA-F and GOA-R have faster convergence speed than GOA-1, GOA-2 and GOA-3 on F14-F19. Because each subpopulation has different form of parameter *c*, GOA-F and GOA-R can simultaneously strengthen the ability of exploration and exploitation in different stages of the optimization process to accelerate convergence.

5.1.2. Sensitivity analysis

In this section, the sensitivity analysis of main parameters is discussed in detail. The size of subpopulation and the number of groups are two main parameters affecting the performance of the multi-group and co-evolution framework. The impact on convergence accuracy and speed of proposed approach is examined by a series of experiments on test functions F1–F19. Table 4 presents the settings of the main parameters for sensitivity analysis. The size of the subpopulation has 5 different scales. The number of groups has 10 levels. Therefore, on each test function, 50 different groups of experiments of GOA-R are run 10 times independently to conduct a comprehensive sensitivity analysis.

Table 4. The settings of the relevant parameters for sensitivity analysis.

Parameters	Parameters range
Size of subpopulation	Five population sizes are set: 10,40,70,90 and 130
Number of groups	Ten different number of groups are set: 1–10

The results can be observed in Figure 5 in detail. The x-axis means the population size, and the numbers 1–5 indicate the population size of 10, 40, 70, 100 and 130 respectively. The y-axis means the number of subpopulations. The z-axis represents the average of the optimal values obtained by each group after 10 independently runs. It should be noted that the results are normalized between 0 and 1 for facilitating the sensitivity analysis. The optimization results will not be significantly improved with the increasing number and size of subpopulations, but sometimes gets worse, such as the results of F2, F5, F8, F11, F15 and F19 in Figure 5. Similar results are also found on other test functions. The search agent of GOA appears to attract each other frequently on the unimodal test functions and have high repulsion rate between each other when settling multi-modal and composite test functions [1]. Therefore, when the size of subpopulation increases, the high repulsion rate will have negative influence on the convergence. Moreover, when the size of population continues to increase, this influence will gradually offset the performance improvement brought by the multi-group and co-evolution framework.

As shown in Figure 6, with the increase in the population size and the number of subpopulations, the running time increases as well on test function F1. The similar results are also found on other test functions. The increase of the number of subpopulations increases the amount of information interaction between subpopulations, and the increase of population size enhances the amount of information interaction within subpopulations.

It can be seen from the above analysis that excessive grouping and large population size can not improve the convergence accuracy, but will significantly increase the running time. A practicable suggestion is that the size of subpopulation should be controlled between 40 and 100, and the number of groups should be controlled between 3 and 5.

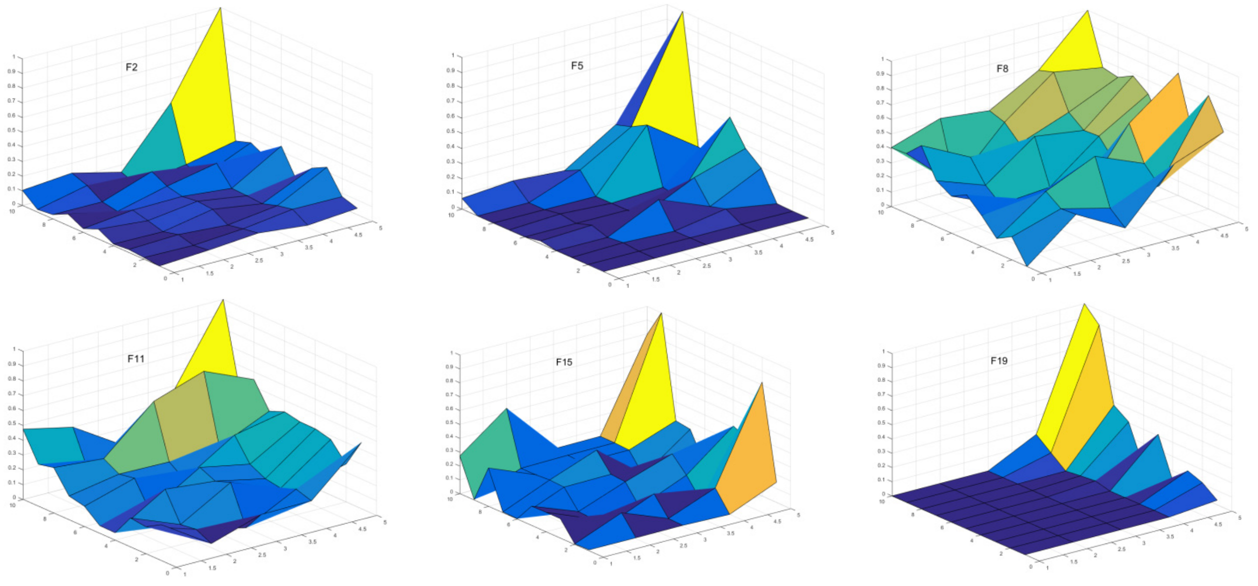

Figure 5. Sensitivity analysis on the population size and the number of subpopulations.

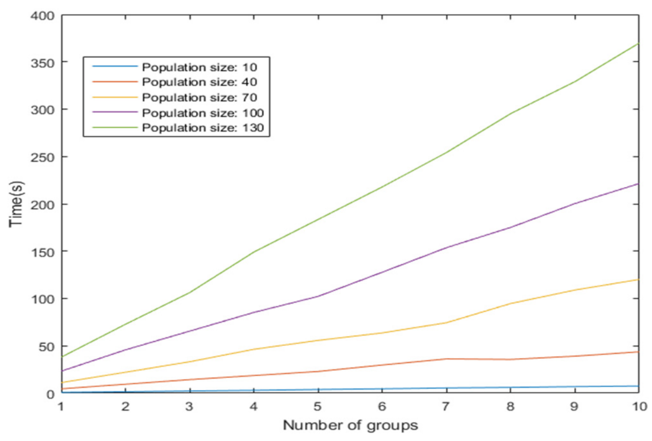

Figure 6. The running time change with the population size and the number of subpopulations.

5.2. Quantitative results and discussion of MOGOA-MC

In this section, several standard multi-objective tests functions with different features are applied to verify the performance of the presented approach in multi-objective optimization. These standard test functions are used in many literatures on multi-objective optimization: ZDT [68], DTLZ [69], and CEC2009 [70].

For the purpose of intuitively assess the performance of MOGOA-MC in convergence, accuracy, and diversity of optimal solutions, two performance indicators are employed: generation distance (GD) and inverse generation distance (IGD) [14]. IGD metric calculates the Euclidean distance between the obtained Pareto optimal solution and the true Pareto optimal solution in the reference set, which can measure the convergence and diversity of the algorithm.

The smaller the value of IGD is, the better the overall performance of the algorithm is.

$$IGD(P,P^*) = \frac{\sum_{x\in P^*} min_{y\in P} dist(x,y)}{|P^*|} \tag{20}$$

where P is the Pareto optimal solution acquired by the algorithm, P^* is a group of uniformly distributed reference points sampled from the true Pareto optimal solutions, and *dist (x, y)* refer to the Euclidean distance.

As a convergence evaluation indicator, GD metric measures the closeness between the obtained Pareto optimal solutions and the true Pareto optimal solutions. The closer GD is to 0, the better the convergence is.

$$GD(P,P^*) = \frac{\sqrt{\sum_{y\in P} min_{x\in P^*} dist(x,y)^2}}{|P|} \tag{21}$$

where the definitions of P, P^*, and *dist (x, y)* are the same as those in IGD.

Similar to GOA-MC, for the purpose of verifying the advantage of the multi-group and co-evolution framework in settling multi-objective optimization problems, we designed a comparative experiment between the MOGOA combined with the multi-group and co-evolution framework (MOGOA-MC) and the original MOGOA. The original MOGOA separately adopts three different forms of parameter c, which are MOGOA-1 using the parameter c in Eq (14), MOGOA-2 using the parameter c in Eq (15) and MOGOA-3 using the parameter c in Eq (16). MOGOA-MC separately adopts two different assignment strategies of parameter c, which are MOGOA-F and MOGOA-R. MOGOA-F adopts fixed assignment strategy to assign a fixed parameter c to each subpopulation from Eq (17). MOGOA-R using random assignment strategy to randomly assign a parameter c for each subpopulation from Eq (18). Other parameters are set uniformly as follows: the amount of population is set to 120, the maximum numbers of iterations is set to 100, the number of subpopulations is 3, c_{max} and c_{min} are set to 1 and 0.00001 respectively.

5.2.1. Results on ZDT and DTLZ

The quantitative results on ZDT and DTLZ are presented in Tables 5–7. All algorithms are run independently 20 times. The average, standard deviation, best, and worst values of IGD are presented in Table 5, and the values of GD are presented in Table 6. The obtained Pareto fronts are qualitatively illustrated in Figures 7 and 8.

Each benchmark test function has its own characteristics, and different settings of key parameters will have an impact on the performance of algorithms. Such as, on ZDT1, ZDT2 and ZDT4, MOGOA-2 utilizing the parameter c with cosine adaption form of NCS have better performance than MOGOA-1 and MOGOA-3; On ZDT3 which has a nonconvex and discontinuous Pareto front, MOGOA-1 utilizing the parameter c with line adaption form of LCS have better performance than MOGOA-2 and MOGOA-3. However, in practical application, it is very difficult to choose the appropriate parameter settings when the optimization problem is not well understood. Therefore, in this paper, in order to solve the problem of parameter selection and setting, a variety of different settings of one key parameter are comprehensively applied in the optimization search process. The effectiveness of the proposed method is also proved by the results in Table 5. Such as, IGD values of MOGOA-F and MOGOA-R are significantly better than MOGOA-1, MOGOA-2 and MOGOA-3 on ZDT1, ZDT3 and ZDT4. On ZDT2, compared with the best algorithm MOGOA-2, MOGOA-F and MOGOA-R also show strong competitiveness.

Table 5. Statistical results for IGD on ZDT1, ZDT2, ZDT3, ZDT4.

IGD	AV.	STD.	Worst	Best	AV.	STD.	Worst	Best
		ZDT1				***ZDT2***		
MOGOA-1	0.008110	4.7486×10^{-5}	0.01781	0.001834	0.02181	9.1174×10^{-5}	0.03179	0.004372
MOGOA-2	0.007148	2.4368×10^{-6}	0.02056	0.002063	**0.005184**	4.7499×10^{-5}	**0.006806**	0.003169
MOGOA-3	0.008415	3.4927×10^{-5}	0.01019	0.002604	0.01787	5.7181×10^{-5}	0.02821	0.003130
MOGOA-F	0.001804	3.0666×10^{-7}	0.003185	0.001253	0.005893	8.6611×10^{-7}	0.02075	**0.001781**
MOGOA-R	**0.001596**	$\mathbf{3.5401 \times 10^{-8}}$	**0.002075**	**0.001159**	0.005196	$\mathbf{7.8079 \times 10^{-7}}$	0.01625	0.001880
		ZDT3				***ZDT4***		
MOGOA-1	0.007790	2.3325×10^{-5}	0.01105	0.003316	0.08460	0.0026	0.1368	0.03810
MOGOA-2	0.01659	1.6409×10^{-5}	0.03471	0.004913	0.1329	0.0024	0.2114	**0.02060**
MOGOA-3	0.02568	1.2805×10^{-6}	0.04807	0.009530	0.1198	0.0048	0.1887	0.03327
MOGOA-F	0.004040	$\mathbf{2.6464 \times 10^{-7}}$	0.008719	**0.002587**	0.06507	0.0020	**0.1041**	0.03486
MOGOA-R	**0.003725**	5.6783×10^{-6}	**0.004526**	0.002669	**0.06345**	**0.0006**	0.1062	0.02554

*Notes: AV.: Average IGD value; STD.: Standard Deviation of IGD obtained from 20 times independently running; Best: the best values of IGD; Worst: the worst values of IGD

Table 6. Statistical results for GD on ZDT1, ZDT2, ZDT3, ZDT4.

GD	Average	STD.	Worst	Best	Average	STD.	Worst	Best
		ZDT1				***ZDT2***		
MOGOA-1	0.01670	2.6674×10^{-5}	0.02259	0.01021	0.03076	6.7313×10^{-5}	0.04413	0.02325
MOGOA-2	0.03256	9.7455×10^{-5}	0.05450	0.01714	0.03358	7.2653×10^{-5}	0.04381	0.01948
MOGOA-3	0.02952	2.2262×10^{-5}	0.04720	0.01451	0.02260	3.3174×10^{-4}	0.03280	0.01217
MOGOA-F	**0.01138**	1.9063×10^{-5}	0.01340	**0.007802**	**0.01892**	5.8046×10^{-5}	0.02858	0.01394
MOGOA-R	0.01207	$\mathbf{6.6201 \times 10^{-6}}$	**0.01498**	0.009695	0.01933	$\mathbf{1.2821 \times 10^{-5}}$	**0.02801**	**0.01195**
		ZDT3				***ZDT4***		
MOGOA-1	0.01762	8.0160×10^{-5}	0.03189	0.008943	0.5279	0.0408	0.8772	0.3554
MOGOA-2	0.01432	1.4615×10^{-4}	**0.01124**	0.01872	0.6291	0.0367	0.8792	0.4846
MOGOA-3	0.02113	1.3262×10^{-5}	0.03587	0.009765	0.5735	0.0095	0.9137	0.3339
MOGOA-F	0.009671	$\mathbf{2.6231 \times 10^{-6}}$	0.01316	0.007496	**0.2822**	0.0101	**0.3790**	0.1958
MOGOA-R	**0.008808**	4.8387×10^{-6}	0.01167	**0.005164**	0.2960	**0.0058**	0.3931	**0.1758**

*Notes: AV.: Average GD value, STD.: Standard Deviation of GD obtained from 20 times independently running; Best: the best values of GD; Worst: the worst values of GD

Through the co-evolution mechanism, the optimization information can be exchanged between subpopulations, so that all search individuals can converge to the true Pareto front as soon as possible. Therefore, the global optimization solutions can be found faster and more accurately. The conclusions can be drawn from the values of GD in Table 6, MOGOA-F and MOGOA-R have significantly better performance than others.

Practical optimization problems usually involve more than two objectives. The more objectives there are, the more complex the optimization problem is, so it is difficult to get its Pareto front and to visualize it. In order to verify the effectiveness of the method in the multi-objective optimization problem with more than two objectives, we benchmark it on DTLZ1. DTLZ1 has three objectives that need to be optimized at the same time, and its Pareto front is a hyperplane. The quantitative results in Table 7 also show the good IGD and GD performance of MOGOA-F and MOGOA-R on DTLZ1. Although the performance improvement is not as good as ZDT series test functions, the effect is still significant.

Table 7. Statistical results for IGD and GD on DTLZ1.

DTLZ1	**IGD**				**GD**			
	AV.	**STD.**	**Worst**	**Best**	**AV.**	**STD.**	**Worst**	**Best**
MOGOA-1	1.5255	0.0059	1.6343	1.4330	8.7203	0.1935	9.1334	8.0797
MOGOA-2	1.4481	0.0096	1.5687	1.3351	10.7424	4.4340	14.1027	8.7297
MOGOA-3	1.4337	0.0171	1.5761	1.2453	10.2828	1.0573	10.2828	8.1977
MOGOA-F	**1.2910**	0.0022	1.3548	**1.2312**	7.1866	0.0513	7.5682	**7.0019**
MOGOA-R	1.3062	**4.7446×10^{-4}**	**1.3297**	1.2826	**7.1640**	**0.0064**	**7.2661**	7.0701

*Notes: AV.: Average IGD or GD value; STD.: Standard Deviation of IGD or GD values obtained from 20 times independently running; Best: the best values of IGD or GD; Worst: the worst values of IGD or GD

The quantitative results in Tables 5–7 verify the superiority of the algorithms with the multi-group and co-evolution framework and prove that the framework can significantly improve the diversity and convergence of Pareto optimal solutions. The qualitative distribution presentation of best Pareto optimal fronts in Figures 7 and 8 also supports this conclusion. In our method, different subpopulations have different settings of key parameters, and the optimization mechanism of search individuals is also different. Therefore, more diverse optimization solutions can be found, and the distribution of optimization schemes is more uniform. The Pareto optimal fronts obtained from MOGOA-F and MOGOA-R are more equally distributed in the true Pareto optimal fronts than that of MOGOA-1, which also prove that the multi-group and co-evolution framework improves the diversity of Pareto optimal solutions.

5.2.2. Results on CEC2009 test functions

In this section, the CEC2009 test functions are applied to further confirm the performance of the proposed methods. UF1–UF7 test functions are bi-objective, UF8–UF10 test functions are tri-objective. The results on CEC2009 test functions are presented in Tables 8 and 9.

The conclusions on CEC2009 test functions are consistent with those of ZDT series test functions and DTLZ test functions. The multi-group mechanism in the multi-group and co-evolution framework gives more differences to the search individuals, which can help to search more diverse optimization solutions; the co-evolution mechanism makes the subpopulations with better performance can transmit information to the whole population, so as to speed up the convergence speed and improve the search efficiency and convergence accuracy.

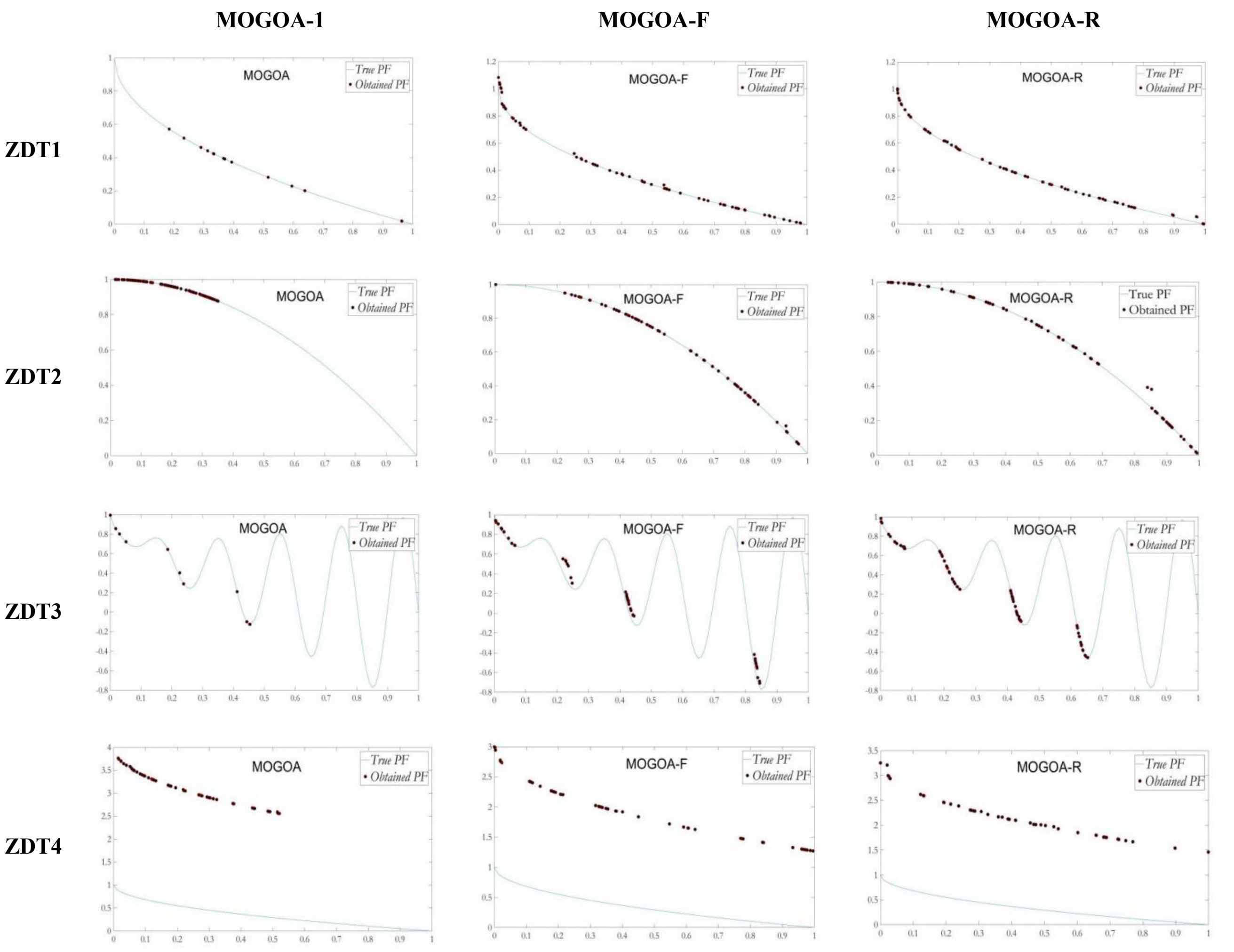

Figure 7. Pareto optimal front on ZDT.

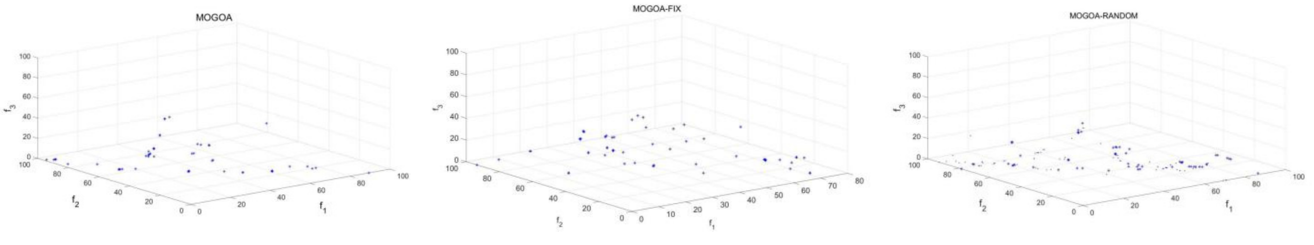

Figure 8. Pareto optimal front on DTLZ1.

Table 8. Statistical results for IGD on UF1 to UF10.

IGD	AV.	STD.	Worst	Best	Average	STD.	Worst	Best
		UF1				***UF2***		
MOGOA-1	0.1151	6.554×10^{-4}	01597	0.0880	0.0594	6.612×10^{-5}	0.0729	0.0468
MOGOA-2	0.0985	1.697×10^{-4}	0.1337	0.0866	0.0645	9.783×10^{-5}	0.0815	0.0463
MOGOA-3	0.1133	9.301×10^{-5}	0.1266	0.0997	0.0816	1.842×10^{-4}	0.1105	0.0635
MOGOA-F	**0.0844**	**1.136×10^{-5}**	**0.0901**	0.0800	0.0439	**1.496** $\times 10^{-5}$	**0.0506**	0.0377
MOGOA-R	0.0865	1.866×10^{-5}	0.0942	**0.0795**	**0.0433**	2.161×10^{-5}	0.0537	**0.0371**
		UF3				***UF4***		
MOGOA-1	0.4206	0.0062	0.5074	0.3117	0.1464	1.478×10^{-4}	0.1617	0.1212
MOGOA-2	0.4534	0.0123	0.6390	0.3373	0.1695	1.623×10^{-3}	0.2355	0.1511
MOGOA-3	0.4240	0.0141	0.5824	0.2429	0.1196	1.118×10^{-4}	0.1306	0.0974
MOGOA-F	**0.3937**	0.0189	0.5836	**0.1989**	0.0979	4.529×10^{-5}	0.1074	**0.0867**
MOGOA-R	0.4129	**0.0111**	**0.5460**	0.2836	**0.0952**	**1.134×10^{-5}**	**0.0998**	0.0903
		UF5				***UF6***		
MOGOA-1	0.6389	0.0181	0.8746	0.4432	0.7147	0.0289	0.9301	0.4911
MOGOA-2	0.7470	0.3465	2.3977	0.3523	0.6317	0.0342	0.9505	0.4775
MOGOA-3	0.7433	0.0369	1.0504	0.4615	0.6590	0.0350	0.8884	0.4050
MOGOA-F	**0.6062**	**0.0152**	**0.7732**	**0.4153**	0.5581	**0.0245**	0.8470	0.3660
MOGOA-R	0.7570	0.0457	1.1485	0.4526	**0.5351**	0.0333	**0.8000**	**0.2680**
		UF7				***UF8***		
MOGOA-1	0.0718	4.958×10^{-5}	0.0856	0.0619	0.2415	4.804×10^{-4}	0.2800	0.2142
MOGOA-2	0.0935	7.649×10^{-4}	0.1363	0.0595	0.2301	7.129×10^{-4}	0.2697	0.1905
MOGOA-3	0.0834	1.507×10^{-4}	0.1066	0.0647	0.2622	8.372×10^{-4}	0.3121	0.2193
MOGOA-F	0.0652	3.215×10^{-5}	**0.0740**	0.0584	**0.1650**	**7.117×10^{-5}**	**0.1785**	**0.1499**
MOGOA-R	**0.0629**	**2.813×10^{-5}**	0.0745	**0.0552**	0.1770	2.142×10^{-4}	0.2077	0.1601
		UF9				***UF10***		
MOGOA-1	0.2485	2.385×10^{-3}	0.3604	0.1980	0.3965	9.223×10^{-3}	0.5928	0.2685
MOGOA-2	0.2381	1.305×10^{-3}	0.3325	0.2174	0.3935	8.569×10^{-3}	0.5442	0.2206
MOGOA-3	0.2874	1.112×10^{-3}	0.3568	0.2414	0.6113	8.244×10^{-3}	1.0740	0.3165
MOGOA-F	0.1836	**2.932×10^{-4}**	0.2122	0.1618	0.2583	6.211×10^{-4}	0.2997	**0.2171**
MOGOA-R	**0.1824**	3.376×10^{-4}	**0.2100**	**0.1515**	**0.2377**	**9.941×10^{-5}**	**0.2552**	0.2254

*Notes: AV.: Average IGD value; STD.: Standard Deviation of IGD obtained from 20 times independently running; Best: the best values of IGD; Worst: the worst values of IGD

The IGD values in Table 8 show that the performance of MOGOA-F and MOGOA-R are also better than that of MOGOA-1, MOGOA-2 and MOGOA-3 on the CEC2009 test suite. The GD values in Table 9 also verify that the proposed algorithms have better convergence than original MOGOA. However, on UF5, UF6 and UF10, although the proposed method is very competitive, it does not show the best performance. Since the Pareto fronts of UF5 and UF6 are discontinuous, and the metric of GD measures the average Euclidean distance from each solution obtained from algorithm to the solution on the nearest true Pareto front, the performance of the proposed methods on this kind of optimization problems are not significant.

In terms of overall performance, these outcomes in Tables 8 and 9 can affirm that the multi-group and co-evolution framework is able to ameliorate the diversity and the convergence of the Pareto optimal solutions.

Table 9. Statistical results for GD on UF1 to UF10.

GD	Average	STD.	Worst	Best	Average	STD.	Worst	Best
		UF1				***UF2***		
MOGOA-1	0.0179	1.359×10^{-4}	0.0439	0.0074	0.0194	5.765×10^{-4}	0.0701	0.0035
MOGOA-2	0.0126	4.841×10^{-5}	0.0254	0.0064	0.0104	1.638×10^{-4}	0.0465	0.0040
MOGOA-3	0.0176	1.815×10^{-4}	0.0516	0.0042	0.0174	2.290×10^{-4}	0.0472	0.0049
MOGOA-F	**0.0081**	$\mathbf{1.557 \times 10^{-5}}$	**0.0163**	**0.0031**	0.0081	$\mathbf{2.046 \times 10^{-5}}$	**0.0149**	**0.0037**
MOGOA-R	0.0118	6.625×10^{-5}	0.0317	0.0033	**0.0080**	6.252×10^{-5}	0.0299	0.0038
		UF3				***UF4***		
MOGOA-1	0.1126	0.0057	0.2702	0.0359	0.0184	2.052×10^{-5}	0.0278	0.0124
MOGOA-2	0.1066	0.0027	0.1804	0.0369	0.0236	9.309×10^{-5}	0.0442	0.0131
MOGOA-3	0.1126	0.0130	0.4048	0.0417	0.0131	2.589×10^{-6}	0.0157	**0.0103**
MOGOA-F	**0.0603**	0.0019	0.1375	**0.0138**	0.0144	8.153×10^{-6}	0.0181	0.0105
MOGOA-R	0.0657	**0.0012**	**0.1187**	0.0298	**0.0130**	$\mathbf{1.071 \times 10^{-6}}$	**0.0153**	0.0119
		UF5				***UF6***		
MOGOA-1	0.1698	0.0018	0.1701	2.046×10^{-4}	0.1313	0.0112	0.3372	**0.0087**
MOGOA-2	0.1545	0.0221	0.5399	0.0307	**0.1056**	**0.0036**	0.2272	0.0149
MOGOA-3	**0.1208**	0.0088	0.2931	$\mathbf{9.984 \times 10^{-13}}$	0.1786	0.0302	0.5893	0.0412
MOGOA-F	0.1299	0.0058	0.3013	0.0198	0.1181	0.0226	**0.2127**	0.0226
MOGOA-R	0.1270	**0.0031**	**0.2334**	0.0773	0.1225	0.0476	0.2446	0.0476
		UF7				***UF8***		
MOGOA-1	0.0074	2.416×10^{-5}	0.0169	0.0013	0.0434	3.596×10^{-4}	0.0693	0.0153
MOGOA-2	0.0083	3.660×10^{-5}	0.0222	0.0019	0.0299	2.001×10^{-4}	0.0571	0.0132
MOGOA-3	0.0084	1.075×10^{-5}	0.0144	0.0034	0.0389	3.710×10^{-4}	0.0691	0.0121
MOGOA-F	0.0051	6.742×10^{-6}	0.0098	**0.0023**	**0.0227**	$\mathbf{6.346 \times 10^{-5}}$	**0.0354**	0.0134
MOGOA-R	**0.0044**	$\mathbf{3.043 \times 10^{-6}}$	**0.0082**	0.0024	0.0289	5.846×10^{-4}	0.0941	**0.0115**
		UF9				***UF10***		
MOGOA-1	0.0459	4.682×10^{-4}	0.0994	0.0245	**0.2103**	6.276×10^{-3}	**0.3394**	0.1107
MOGOA-2	0.0455	1.671×10^{-4}	0.0643	0.0259	0.2502	$\mathbf{1.831 \times 10^{-3}}$	0.5226	**0.0447**
MOGOA-3	0.0612	1.127×10^{-3}	0.1237	**0.0162**	0.2603	2.171×10^{-3}	0.5838	0.0926
MOGOA-F	**0.0374**	$\mathbf{1.021 \times 10^{-4}}$	**0.0523**	0.0232	0.2697	3.392×10^{-3}	0.3572	0.1663
MOGOA-R	0.0424	2.216×10^{-4}	0.0659	0.0166	0.2306	3.232×10^{-3}	0.3497	0.1622

*Notes: AV.: Average GD value; STD.: Standard Deviation of GD obtained from 20 times independently running; Best: the best values of GD; Worst: the worst values of GD

The average, standard deviation, maximum, and minimum in the previous tables are obtained from the results of 20 independent runs. They reflect the average performance of the algorithm. To show that the outcomes were not acquired accidentally, the Wilcoxon rank-sum test is employed. The p-values are the results of the Wilcoxon rank-sum test. When p-values are less than 0.05, it shows that rank sum rejects the null hypothesis of equal medians at the default 5% significance level. The higher the p value, the more competitive the algorithm is, even if it is not the best one. The p-values of the algorithm with the best average performance in each test function is "N/A". Then the best performing algorithm is chosen to test with other algorithms in pairs. The p-values in Table 10 show that MOGOA-F and MOGOA-R based on the multi-group and co-evolution framework performs well on most of the test functions. The p-values between MOGOA-F and MOGOA-R are all greater than 0.05 on GD and IGD, which indicates that the two algorithms are highly competitive with each other. While MOGOA-1, MOGOA-2 or MOGOA-3 are not the algorithm with the best average performance, their p-values are less than 0.05 in most test functions, so they are not competitive.

Table 10. P-values acquired from the rank sum test on UF1 to UF10.

Test functions	GD					IGD				
	M-1	**M-2**	**M-3**	**M-F**	**M-R**	**M-1**	**M-2**	**M-3**	**M-F**	**M-R**
UF1	0.0073	0.1212	0.0211	**N/A**	0.3075	0.0001	0.0001	0.0001	**N/A**	0.3075
UF2	0.2730	0.4274	0.0257	0.9698	**N/A**	0.0001	0.0001	0.0001	0.6232	**N/A**
UF3	0.0757	0.0376	0.1405	**N/A**	0.6232	0.6776	0.3447	0.6232	**N/A**	0.7337
UF4	0.0017	0.0001	0.5205	0.2730	N/A	0.0001	0.0001	0.0001	0.1212	**N/A**
UF5	0.1620	0.9698	**N/A**	0.6776	0.7337	0.6232	0.7913	0.1212	**N/A**	0.1041
UF6	0.9097	**N/A**	0.4727	0.9097	0.6232	0.0452	0.3847	0.1859	0.9097	**N/A**
UF7	0.0922	0.0708	0.0036	0.8501	**N/A**	0.0073	0.0058	0.0001	0.5205	**N/A**
UF8	0.0140	0.0166	0.0241	**N/A**	**0.8501**	0.0001	0.0001	0.0001	**N/A**	0.0376
UF9	0.2725	0.1315	0.0257	**N/A**	0.3881	0.0008	0.0001	0.0001	0.9097	**N/A**
UF10	**N/A**	0.4272	0.6232	0.1041	0.3075	0.0001	0.0028	0.0001	0.0211	**N/A**

*Notes: M-1: MOGOA-1; M-2: MOGOA-2; M-3: MOGOA-3; M-F: MOGOA-F; M-R: MOGOA-R

In general, the above experimental outcomes indicate that the improvement of the multi-group and co-evolution framework to the diversity and convergence of MOGOA is very significant. Although MOGOA has proved its superiority in reference [14], which are compared with other competitive multi-objective optimization algorithms, such as MOPSO, MOEA/D. The proposed methods are greatly effective for settling optimization problems with multiple objectives. This is mainly attributed to these two features of the framework: firstly, the grouping of subpopulations with different evolutionary mechanisms increases the diversity of search agents and strengthens the exploration capability of the optimization algorithm; secondly, the co-evolution mechanism between subpopulations ensures the convergence of the algorithm, and accelerates the convergence speed.

6. Conclusions

In this paper, a multi-group and co-evolution framework for meta-heuristic methods is proposed, which is employed to improve the performance of GOA. With the fast convergence and high accuracy, the promising performance of the framework is verified on the benchmark test functions. The sensitivity of the main parameters is analyzed to check the feasibility and validation of the proposed framework. For the multi-objective optimization problems, the proposed framework was used to extend the MOGOA algorithm. To evaluate the performance improvement brought by the proposed framework, the numerical evaluations were also conducted by comparing with the original MOGOA on a series of test suits. GD and IGD were calculated to prove the advancement of diversity and accuracy of solutions by the proposed framework. Moreover, the Wilcoxon rank-sum tests were also conducted to show the statistically significant. It could be said that these improvements were due to the modifications in the difference and interaction of search agents, which led to more convenient exploration and more active exploitation. The proposed methods in this paper can find a fine balance between exploration and exploitation, which is particularly important in solving multimodal search space and large-scale optimization problems without deep understanding, such as feature selection and neural network training in machine learning, job-shop scheduling and control of power systems in engineering applications. Experimental results show the effectiveness of the proposed method, but there are still some limitations. Because of no free lunch theorem in the optimization domain, the proposed method needs further adjustment and modification to adapt to the actual specific problems, such as binary, dynamic, discrete, and others. Due to the existence of co-evolution among subpopulations, the proposed method spends more time than original algorithm in the optimization process. The multi-group and co-evolution framework is only applicable to the same kind of swarm intelligence algorithm at present. Therefore, for future research, it is suggested to further improve the multi-group and co-evolution framework for integrating various swarm intelligence algorithms. The settings of key parameters should also be quantitatively explained according to specific problems in this framework. It will be focused on the applications on more complex engineering problems.

Acknowledgments

This work is partially supported by the Natural Science Foundation of China under Grant (31771679, 31671589), the Anhui Foundation for Science and Technology Major Project, China, under Grant (16030701092,18030901034, 201904e01020006), the Key Laboratory of Agricultural Electronic, Commerce, Ministry of Agriculture of China under Grant (AEC2018003, AEC2018006), the 2019 Anhui University collaborative innovation project (GXXT-2019-013), the Hefei Major Research Project of Key Technology (J2018G14), Natural Science Research Project of Anhui Provincial Department of Education (KJ2020A0107).

Conflicts of interest

The authors declare no conflict of interest.

References

1. S. Saremi, S. Mirjalili, A. Lewis, Grasshopper Optimisation Algorithm: Theory and application, *Adv. Eng. Software*, **105** (2017), 30–47.
2. M. Laszczyk, P. B. Myszkowski, Improved selection in evolutionary multi–objective optimization of Multi–Skill Resource–Constrained Project Scheduling Problem, *Inf. Sci.*, **481** (2019), 412–431.

3. G. Eichfelder, J. Niebling, S. Rocktäschel, An algorithmic approach to multi-objective optimization with decision uncertainty, *J. Global Optim.,* **77** (2020), 3–25.
4. D. Simon, *Evolutionary Optimization Algorithms*, John Wiley & Sons, Hoboken, NJ, 2013.
5. K. Deb, Multi-objective optimization, Search Methodologies, *Search Methodol.*, **2014** (2014), 403–449.
6. T. Eftimov, P. Koroec, Deep Statistical Comparison for Multi-Objective Stochastic Optimization Algorithms, *Swarm Evol. Comput.*, **61** (2020), 100837.
7. G. G. Wang, S. Deb, Z. Cui, Monarch butterfly optimization, *Neural Comput. Appl.*, **31** (2015), 1995–2014.
8. S. Li, H. Chen, M. Wang, A. A. Heidari, S. Mirjalili, Slime mould algorithm: A new method for stochastic optimization, *Future Gener. Comput. Syst.*, **111** (2020), 300–323.
9. G. G. Wang, Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems, *Memetic Comput.*, **10** (2018), 151–164
10. A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: Algorithm and applications, *Future Gener. Comput. Syst.*, **97** (2019), 849–872.
11. L. Lin, M. Gen, Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation, *Soft Comput.*, **13** (2009), 157–168.
12. X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, 2010.
13. I. Boussad, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, *Inf. Sci.*, **237** (2013), 82–117.
14. S. Z. Mirjalili, S. Mirjalili, S. Saremi, H. Faris, I. Aljarah, Grasshopper optimization algorithm for multi-objective optimization problems, *Appl. Intell.*, **48** (2018), 805–820.
15. K. Deb, S. Mittal, D. K. Saxena, E. D. Goodman, *Embedding a Repair Operator in Evolutionary Single and Multi-Objective Algorithms-An Exploitation-Exploration Perspective*, Evolutionary Multi-Criterion Optimization (EMO-2021), 2021.
16. D. Li, W. Guo, A. Lerch, Y. Li, L. Wang, Q. Wu, An adaptive particle swarm optimizer with decoupled exploration and exploitation for large scale optimization, *Swarm Evol. Comput.*, **60** (2021), 100789.
17. M. Abdel-Basset, R. Mohamed, M. Abouhawwash, Balanced multi-objective optimization algorithm using improvement based reference points approach, *Swarm Evol. Comput.*, **60** (2021), 100791.
18. H. Zhang, J. Sun, T. Liu, K. Zhang, Q. Zhang, Balancing Exploration and Exploitation in Multi-objective Evolutionary Optimization, *Inf. Sci.*, **497** (2019), 129–148.
19. P. Koroec, T. Eftimov, Insights into Exploration and Exploitation Power of Optimization Algorithm Using DSC Tool, *Mathematics*, **8** (2020), 1474–1484.
20. S. H. Liu, M. Mernik, B. R. Bryant, To explore or to exploit: An entropy-driven approach for evolutionary algorithms, *Int. J. Knowl. Based Intell. Eng. Syst.*, **13** (2009), 185–206.
21. J. J. Liang, P. N. Suganthan, *Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constraint-Handling Mechanism*, IEEE International Conference on Evolutionary Computation, IEEE, 2006.
22. X. Wu, S. Zhang, W. Xiao, Y. Yin, The Exploration/Exploitation Tradeoff in Whale Optimization Algorithm, *IEEE Access*, **7** (2019), 125919–125928.
23. T. Jiang, C. Zhang, H. Zhu, J. Gu, G. Deng, Energy-efficient scheduling for a job shop using an improved whale optimization algorithm, *Mathematics*, **6** (2018), 220–236.

24. J. Too, A. R. Abdullah, N. M. Saad, A New Co-Evolution Binary Particle Swarm Optimization with Multiple Inertia Weight Strategy for Feature Selection, *Informatics*, **6** (2019), 21–34.
25. L. Abualigah, A. Diabat, A comprehensive survey of the Grasshopper optimization algorithm: results, variants, and applications, *Neural Comput. Appl.*, **32** (2020), 15533–15556.
26. R. A. Ibrahim, A. A. Ewees, D. Oliva, M. A. Elaziz, S. Lu, Improved salp swarm algorithm based on particle swarm optimization for feature selection, *J. Ambient Intell. Humanized Comput.*, **10** (2018), 3155–3169.
27. K. Li, S. Kwong, Q. Zhang, K. Deb, Interrelationship-Based Selection for Decomposition Multiobjective Optimization, *IEEE Trans. Cybern.*, **45** (2015), 2076–2088.
28. S. W. Jiang, Z. H. Cai, J. Zhang, Y. S. Ong, *Multiobjective optimization bydecomposition with Pareto-adaptive weight vectors*, Seventh International Conference on Natural Computation, IEEE, 2011.
29. S. Shahbeig, A. Rahideh, M. S. Helfroush, K. Kazemi, Gene selection from large-scale gene expression data based on fuzzy interactive multi-objective binary optimization for medical diagnosis, *Biocybern. Biomed. Eng.*, **38** (2018), 313–328.
30. S. Kim, I. J. Jeong, *Interactive Multi-Objective Optimization Using Mobile Application: Application to Multi-Objective Linear Assignment Problem*, Proceedings of the 2019 Asia Pacific Information Technology Conference, 2019.
31. M. Sakawa, Fuzzy Multiobjective Optimization, in *Multicriteria Decision Aid and Artificial Intelligence*, John Wiley & Sons, Ltd, 2013, 235–271.
32. Y. Tian, S. Yang, X. Zhang, An Evolutionary Multi-objective Optimization Based Fuzzy Method for Overlapping Community Detection, *IEEE Trans. Fuzzy Syst.*, **28** (2019), 2841–2855.
33. N. Piegay, D. Breysse, Multi-Objective Optimization and Decision Aid for Spread Footing Design in Uncertain Environment, in *Geotechnical Safety and Risk V*, IOS Press, 2015, 419–424.
34. E. D. Comanita, C. Ghinea, R. M. Hlihor, I. M. Simion, C. Smaranda, L. Favier, et al., Challenges and opportunities in green plastics: an assessment using the electre decision-aid method, *Environ. Eng. Manage. J.*, **14** (2015), 689–702.
35. J. Zhou, X. Yao, L. Gao, C. Hu, An indicator and adaptive region division based evolutionary algorithm for many-objective optimization, *Appl. Soft Comput.*, **99** (2021), 106872.
36. H. Zhang, Q. Hui, Many objective cooperative bat searching algorithm, *Appl. Soft Comput.*, **77** (2019), 412–437.
37. I. H. Osman, G. Laporte, Metaheuristics: A bibliography, *Ann. Oper. Res.*, **63** (1996), 511–623.
38. S. Santander-Jiménez, M. A. Vega-Rodríguez, L. Sousa, A multiobjective adaptive approach for the inference of evolutionary relationships in protein-based scenarios, *Inf. Sci.*, **485** (2019), 281–300.
39. B. Yazid, B. Sadek, C. Djamal, Evolutionary Metaheuristics to Solve Multiobjective Assignment Problem in Telecommunication Network: Multiobjective Assignment Problem, *Int. J. Appl. Metaheuristic Comput.*, **11** (2020), 56–76.
40. Á. Rubio-Largo, L. Vanneschi, M. Castelli, M. A. Vega-Rodríguez, Multiobjective Meta-heuristic to Design RNA Sequences, *IEEE Trans. Evol. Comput.*, **23** (2018), 156–169.
41. S. Safarzadeh, S. Shadrokh, A. Salehian, A heuristic scheduling method for the pipe-spool fabrication process, *J. Ambient Intell. Humanized Comput.,* **9** (2018), 1901–1918.
42. S. Safarzadeh, H. Koosha, Solving an extended multi-row facility layout problem with fuzzy clearances using GA, *Appl. Soft Comput.*, **61** (2017), 819–831.

43. C. M. Rahman, T. A. Rashid, A new evolutionary algorithm: Learner performance based behavior algorithm, *Egypt. Inf. J.*, (2020), forthcoming.
44. C. M. Rahman, T. A. Rashid, Dragonfly Algorithm and its Applications in Applied Science Survey, *Comput. Intell. Neurosci.*, **2019** (2019), 9293617.
45. A. M. Ahmed, T. A. Rashid, S. M. Saeed, Cat Swarm Optimization Algorithm: A Survey and Performance Evaluation, *Comput. Intell. Neurosci.*, **2020** (2020), 20.
46. B. A. Hassan, T. A. Rashid, Operational Framework for Recent Advances in Backtracking Search Optimisation Algorithm: A Systematic Review and Performance Evaluation, *Appl. Math. Comput.*, **370** (2019), 124919.
47. A. S. Shamsaldin, T. A. Rashid, R. A. Al-Rashid, N. K. Al-Salihi, M. Mohammadi, Donkey and Smuggler Optimization Algorithm: A Collaborative Working Approach to Path Finding, *J. Comput. Design Eng.*, **6** (2019), 562–583.
48. J. M. Abdullah, T. Rashid, Fitness Dependent Optimizer: Inspired by the Bee Swarming Reproductive Process, *IEEE Access*, **7** (2019), 43473–43486.
49. D. A. Muhammed, S. A. M. Saeed, T. A. Rashid, Improved Fitness-Dependent Optimizer Algorithm, *IEEE Access*, **8** (2020), 19074–19088.
50. M. A. Montes, T. Stutzle, M. Birattari, M. Dorigo, Frankenstein's PSO: A Composite Particle Swarm Optimization Algorithm, *IEEE Trans. Evol. Comput.*, **13** (2009), 1120–1132.
51. Q. T. Vien, T. A. Le, X. S. Yang, T. Q. Duong, Enhancing Security of MME Handover via Fractional Programming and Firefly Algorithm, *IEEE Trans. Commun.*, **67** (2019), 6206–6220.
52. G. J. Ibrahim, T. A. Rashid, M. O. Akinsolu, An energy efficient service composition mechanism using a hybrid meta-heuristic algorithm in a mobile cloud environment, *J. Parallel Distrib. Comput.*, **143** (2020), 77–87.
53. H. Mohammed, T. A. Rashid, A Novel Hybrid GWO with WOA for Global Numerical Optimization and Solving Pressure Vessel Design, *Neural Comput. Appl.*, **32** (2020), 14701–14718.
54. H. M. Mohammed, S. U. Umar, T. A. Rashid, A Systematic and Meta-analysis Survey of Whale Optimization Algorithm, *Comput. Intell. Neurosci.*, **2019** (2019), 8718571.
55. P. Sharma, A. Gupta, A. Aggarwal, D. Gupta, A. Khanna, A. E. Hassanien, The health of things for classification of protein structure using improved grey wolf optimization, *J. Supercomput.*, **76** (2020), 1226–1241.
56. H. Zhang, S. Su, A hybrid multi-agent Coordination Optimization Algorithm, *Swarm Evol. Comput.*, **51** (2019), 100603.
57. H. Jia, Y. Li, C. Lang, X. Peng, K. Sun, J. Li, Hybrid grasshopper optimization algorithm and differential evolution for global optimization, *J. Intell. Fuzzy Syst.*, **37** (2019), 6899–6910.
58. Q. Lin, Q. Zhu, P. Huang, J. Chen, Z. Ming, J. Yu, A novel hybrid multi-objective immune algorithm with adaptive differential evolution, *Comput. Oper. Res.*, **62** (2015), 95–111.
59. S. Arora, P. Anand, Chaotic grasshopper optimization algorithm for global optimization, *Neural Comput. Appl.*, **31** (2019), 4385–4405.
60. Z. Elmi, M. Ö. Efe, *Multi-objective grasshopper optimization algorithm for robot path planning in static environments,* 2018 IEEE International Conference on Industrial Technology, 2018.
61. S. Rangasamy, Y. Kuppusami, A Novel Nature-Inspired Improved Grasshopper Optimization-Tuned Dual-Input Controller for Enhancing Stability of Interconnected Systems, *J. Circuits Syst. Comput.*, **2020** (2020), 21501346.

62. X. Zhang, Q. Miao, H. Zhang, L. Wang, A parameter-adaptive VMD method based on grasshopper optimization algorithm to analyze vibration signals from rotating machinery, *Mech. Syst. Signal Process.*, **108** (2018), 58–72.
63. S. Dwivedi, M. Vardhan, S. Tripathi, Building an efficient intrusion detection system using grasshopper optimization algorithm for anomaly detection, *Cluster Comput.*, **2021** (2021),1–20.
64. Y. Li, L. Gu, Grasshopper optimization algorithm based on curve adaptive and simulated annealing, *Appl. Res. Comput.*, **36** (2019), 3637–3643.
65. R. Yaghobzadeh, S. R. Kamel, M. Asgari, H. Saadatmand, A Binary Grasshopper Optimization Algorithm for Feature Selection, *Int. J. Eng. Res. Technol.*, **9** (2020), 533–540.
66. S. Dwivedi, M. Vardhan, S. Tripathi, An Effect of Chaos Grasshopper Optimization Algorithm for Protection of Network Infrastructure, *Comput. Networks*, **176** (2020), 107251.
67. R. V. Rao, *Application of TLBO and ETLBO Algorithms on Complex Composite Test Functions*, Teaching Learning Based Optimization Algorithm, 2016.
68. E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: Empirical results, *Evol. Comput.*, **8** (2000), 173–195.
69. K. Deb, L. Thiele, M. Laumanns, E. Zitzler, *Scalable multi-objective optimization test problems*, Proceedings of the 2002 congress on evolutionary computation, 2002.
70. Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, S. Tiwari, Multiobjective optimization test instances for the CEC 2009 special session and competition, University of Essex, Colchester, UK and Nanyang Technological University, *Rep. CES*, **487** (2008), 2008.

Appendix

Test functions utilized in this paper.

Table A1. Unimodal benchmark functions.

Function	Dim	Range	$f_{\min}$
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	[–100,100]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[–10,10]	0
$f_3(x) = \sum_{i=1}^{i} (\sum_{j\ 1}^{i} x_j)^2$	30	[–100,100]	0
$f_4 = \max\{ \lvert x_i \rvert, 1 \le i \le n \}$	30	[–100,100]	0
$f_5(x) = \sum_{i=1}^{n\ 1} [100\,(x_{i+1} \quad x_i^2)^2 + (x_i \quad 1)^2]$	30	[–30,30]	0
$f_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	[–100,100]	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 random\ [0,1)$	30	[–128,128]	0

Table A2. Multimodal benchmark functions.

Function	Dim	Range	$f_{\min}$
$F_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	[–500,500]	–418.9829 × Dim
$F_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	30	[–5.12,5.12]	0
$F_{10}(x) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e$	30	[–32,32]	0
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[–600,600]	0
$F_{12}(x) = \frac{\pi}{n}\left\{10(sin\,\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\}$ $+ \sum_{i=1}^{n} u(x_i, 10,100,4) + \sum_{i=1}^{n} u(x_i, 10,100,4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m x_i > a \\ 0 - a < x_i < a \\ k(-x_i - a)^m x_i < -a \end{cases}$	30	[–50,50]	0
$F_{13}(x) = 0.1\left\{sin^2(3\pi x_1)\right.$ $+ \sum_{i=1}^{n}(x_1 - 1)^2[1 + sin^2(3\pi x_i + 1)]$ $\left. + (x_n - 1)^2[1 + sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 10,100,4)$	30	[–50,50]	0

Table A3. Composite benchmark functions.

Function	Dim	Range	$f_{\min}$
$F_{14}(CF1)$: $f_1, f_2, f_3, ..., f_{10} = \textit{Sphere Function}$ $[\sigma_1, \sigma_2, \sigma_3, ..., \sigma_{10}] = [1,1,1,...,1]$ $[\lambda_1, \lambda_2, \lambda_3, ..., \lambda_{10}] = [5/100, 5/100, 5/100, ..., 5/100]$	30	[–5,5]	0
$F_{15}(CF2)$: $f_1, f_2, f_3, ..., f_{10} = \textit{Griewank's Function}$ $[\sigma_1, \sigma_2, \sigma_3, ..., \sigma_{10}] = [1,1,1,...,1]$ $[\lambda_1, \lambda_2, \lambda_3, ..., \lambda_{10}] = [5/100, 5/100, 5/100, ..., 5/100]$	30	[–5,5]	0
$F_{16}(CF3)$: $f_1, f_2, f_3, ..., f_{10} = \textit{Griewank's Function}$ $[\sigma_1, \sigma_2, \sigma_3, ..., \sigma_{10}] = [1,1,1,...,1]$ $[\lambda_1, \lambda_2, \lambda_3, ..., \lambda_{10}] = [1,1,1,...,1]$	30	[–5,5]	0
$f_{17}(CF4)$: $f_1, f_2 = \textit{Ackley's Function}$ $f_3, f_4 = \textit{Rastrigin's Function}$ $f_5, f_6 = \textit{Weierstrass's Function}$ $f_7, f_8 = \textit{Griewank's Function}$ $f_9, f_{10} = \textit{Sphere Function}$ $[\sigma_1, \sigma_2, \sigma_3, ..., \sigma_{10}] = [1,1,1,...,1]$ $[\lambda_1, \lambda_2, \lambda_3, ..., \lambda_{10}] = [5/32, 5/32, 1, 1, 5/0.5, 5/0.5, 5/100, 5/100, 5/100, 5/100]$	30	[–5,5]	0
$f_{18}(CF5)$: $f_1, f_2 = \textit{Rastrigin's Function}$ $f_3, f_4 = \textit{Weierstrass's Function}$ $f_5, f_6 = \textit{Griewank's Function}$ $f_7, f_8 = \textit{Ackley's Function}$ $f_9, f_{10} = \textit{Sphere Function}$ $[\sigma_1, \sigma_2, \sigma_3, ..., \sigma_{10}] = [1,1,1,...,1]$ $[\lambda_1, \lambda_2, \lambda_3, ..., \lambda_{10}] = [1/5, 1/5, 5/0.5, 5/0.5, 5/100, 5/100, 5/32, 5/32, 5/100, 5/100]$	30	[–5,5]	0
$f_{19}(CF6)$: $f_1, f_2 = \textit{Rastrigin's Function}$ $f_3, f_4 = \textit{Weierstrass's Function}$ $f_5, f_6 = \textit{Griewank's Function}$ $f_7, f_8 = \textit{Ackley's Function}$ $f_9, f_{10} = \textit{Sphere Function}$ $[\sigma_1, \sigma_2, \sigma_3, ..., \sigma_{10}] = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$ $[\lambda_1, \lambda_2, \lambda_3, ..., \lambda_{10}] = [0.1*1/5, 0.2*1/5, 0.3*5/0.5, 0.4*5/0.5, 0.5*5/100]$ $0.6*5/100, 0.7*5/32, 0.8*5/32, 0.9*5/100, 1*5/100$	30	[–5,5]	0

Table A4. Multi-objective benchmark functions (ZDT & DTLZ1).

Name	Function
ZDT1	$\min f_1(x_1) = x_1$ $\min f_2(x) = g(1 - \sqrt{(f_1/g)})$ $g(x) = 1 + 9 \sum_{i=2}^{m} x_1 /(m - 1)$ $s.t. 0 \le x_i \le 1, i = 1,2,..., 30$
ZDT2	$\min f_1(x_1) = x_1$ $\min f_2(x) = g(1 \quad (f_1/g)^2)$ $g(x) = 1 + 9 \sum_{i=2}^{m} x_i /(m \quad 1)$ $\text{s.t.} 0 \le x_i \le 1, i = 1,2,...,30$
ZDT3	$\min f_1(x_1) = x_1$ $\min f_2(x) = g(1 \quad \sqrt{f_1/g} \quad (f_1/g)\sin(10\pi f_1))$ $g(x) = 1 + 9 \sum_{i=2}^{m} x_i /(m \quad 1)$ $s.t. 0 \le x_i \le 1, i = 1,2,...,30$
ZDT4	$\min f_1(x_1) = x_1$ $\min f_2(x) = g(1 \quad \sqrt{(f_1/g)})$ $g(x) = 1 + 10(n \quad 1) + \sum_{i=2}^{m} (x_i \quad 10\cos(4\pi x_i))$ $s.t. 0 \le x_1 \le 1, -5 \le x_i \le 5, i = 2,...,9$
DTLZ1	$\min f_1(x_1) = 0.5 x_1 x_2 (1 + g)$ $\min f_2(x) = 0.5 x_1 (1 \quad x_2)(1 + g)$ $\min f_3(x) = 0.5(1 \quad x_1)(1 + g)$ $g(x) = 100(5 + \sum_{i=3}^{m} (x_i \quad 0.5))^2 \quad \cos(20\pi x_i \quad 0.5)$ $0 \le x_i \le 1, i = 1,2,...,30$

Table A5. Bi-objective test problems (CEC2009).

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

Table A6. Tri-objective test problems (CEC2009).

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