Hu *et al. J Smart Environ Green Comput* 2024;4:4-21
DOI: 10.20517/jsegc.2023.18

Journal of Smart Environments and Green Computing

Original Article

Open Access

A cluster-based co-evolutionary optimization method for bilevel multi-objective optimization

Wan-Yue Hu[1], Fei Li[1], Pei-Qiu Huang[2], Er-Qian Ge[1]

[1]School of Electrical and Information Engineering, Anhui University of Technology, Ma'anshan, 243002, China.
[2]School of Automation, Central South University, Changsha 410083, China.

Correspondence to: Prof. Fei Li, School of Electrical and Information Engineering, Anhui University of Technology, No. 1530 Ma Xiang Road, Ma'anshan 243002, Anhui, China. E-mail: lanceleeneu@126.com

How to cite this article: Hu WY, Li F, Huang PQ, Ge EQ. A cluster-based co-evolutionary optimization method for bilevel multi-objective optimization. *J Smart Environ Green Comput* 2024;4:4-21. http://dx.doi.org/10.20517/jsegc.2023.018

Received: 3 Aug 2023 **First Decision:** 15 Nov 2023 **Revised:** 29 Nov 2023 **Accepted:** 14 Dec 2023 **Published:** 5 Jan 2024

Academic Editors: Witold Pedrycz, Ferdinando Di Martino **Copy Editor:** Dong-Li Li **Production Editor:** Dong-Li Li

Abstract

The research motivation of multi-objective bilevel optimization mainly stems from the need to solve practical problems and improve decision-making efficiency. On the one hand, bilevel optimization helps to solve the complexity and uncertainty in real life, thereby improving decision-making efficiency and robustness. On the other hand, by promoting the development and application of AI technology, bilevel optimization also provides support for sustainable development. Although the application of bilevel optimization has proven to be beneficial in addressing various real-life problems. However, recent studies indicate that achieving both high speed and high-quality optimization through existing algorithms remains challenging. This difficulty arises due to the NP-hard nature of the bilevel optimization problem. The nested structure method, commonly used to tackle this problem, involves each upper level solution independently performing the lower level optimization task. This approach significantly increases the number of evaluations for the lower level. To address this issue, our proposed method leverages the similarity in lower level optimization to group upper level solutions, enabling co-evolution of lower level solutions within the same group. Consequently, this approach substantially reduces the number of evaluations required for lower level solutions. Additionally, our method pairs parents and offspring, the optimized lower level solutions of the parents are utilized to optimize the lower level solutions of the offspring. This approach accelerates the optimization process for the lower level. To validate the effectiveness of our algorithm, we have applied it to a suite of test problems, demonstrating satisfactory performance.

Keywords: Bilevel multi-objective optimization, multi-task optimization, evolutionary algorithms, multi-objective optimization

1. INTRODUCTION

Today's society faces two major challenges: sustainable development and optimal use of resources. Green computing aims to reduce the environmental impact of computers and information technology, including reducing energy consumption and reducing e-waste. Bilevel multi-objective optimisation is an optimisation framework that considers multiple objective functions at the same time to find a good balance. By incorporating green computing principles, such as minimizing energy consumption and reducing environmental impact, as additional objectives or constraints within the optimization framework, we can effectively prioritize sustainability alongside other performance metrics. The bilevel optimization problem (BLOP) is a nested structural problem, where a lower level optimization problem is embedded within the upper level optimization problem [1]. Bilevel optimization has its roots in game theory [2]. It has gained increased attention due to its potential usage in various fields, such as economic management, resource allocation, energy sustainability, transportation planning, machine learning and medical engineering [3–9]. In some cases, both the upper level and lower level optimization problems may contain multiple conflicting objective functions, leading to multi-objective bilevel optimization problems (MBOPs). Without loss of generality, a MBOP can be expressed as follows:

$$\begin{aligned} \min_{\mathbf{x}^u,\mathbf{x}^l} \ & \mathbf{f}^u = \{f_1^u(\mathbf{x}^u,\mathbf{x}^l),\ldots,f_{n^u}^u(\mathbf{x}^u,\mathbf{x}^l)\}, \\ \text{s.t. } & g_j^u(\mathbf{x}^u,\mathbf{x}^l) \le 0, j=1,\ldots,p^u, \\ & \mathbf{x}^l \in \arg\min_{\mathbf{x}^l} \mathbf{f}^l = \{f_1^l(\mathbf{x}^u,\mathbf{x}^l),\ldots,f_{n^l}^l(\mathbf{x}^u,\mathbf{x}^l)\}, \\ & \text{s.t. } g_j^l(\mathbf{x}^u,\mathbf{x}^l) \le 0, j=1,\ldots,p^l, \end{aligned} \tag{1}$$

where $\mathbf{f}^u$ and $\mathbf{f}^l$ are the upper level and lower level objective functions, respectively; $\mathbf{x}^u$ and $\mathbf{x}^l$ refer to the upper level and lower level solutions, respectively; n^u and n^l indicate the number of upper level and lower level objective functions, respectively; $g_j^u(\mathbf{x}^u,\mathbf{x}^l)$ and $g_j^l(\mathbf{x}^u,\mathbf{x}^l)$ are the jth upper level and lower level constraints, respectively; and p^u and p^l denote the number of upper level and lower level constraints, respectively. If and only if the upper level constraints are satisfied and $\mathbf{x}^l$ is a Pareto-optimal solution to the lower level optimization problem with regard to the given $\mathbf{x}^u$, then a solution $\mathbf{x} = (\mathbf{x}^u,\mathbf{x}^l)$ is a feasible solution of a BMOP. The goal of solving a BMOP is to find a set of widely distributed feasible solutions with respect to the upper level optimization problem. Due to the nested structure, traditional optimization methods usually have difficulty solving MBOPs effectively. Evolutionary algorithms (EAs) have been used to solve MBOPs [10,11] because they do not depend on the mathematical characteristics of MBOPs. The existing EAs for solving MBOPs can be classified into three types: single-level reduction methods [12–17], surrogate-model-based methods [6,18,19], and nested-based methods [10,20–22].

The single-level reduction approach converts a MBOP into a single-level optimization problem and then applies EAs to solve it. For instance, Li *et al.* used adaptive weighting sum scalarization and Karush-Kuhn-Tucker (KKT) conditions to transform a BMOP to a multi-objective optimization problem (MOP) [12]. Then they proposed an effective smoothing technique to cope with complementarity constraints. Li *et al.* [14] improved the algorithm presented in [12]. They first used the KKT condition to transform a BMOP into a MOP involving complementarity constraints and then proposed a decomposition-based constrained multi-objective differential EA. Jia *et al.* converted a MBOP into a MOP based on the primal and dual theory and used multi-objective metaheuristics and constraint processing techniques for optimization [13]. The method of converting a BMOP into a single-level optimization problem is very mature in the field of bilevel optimization, but these methods usually require strict mathematical assumptions, and these assumptions are usually difficult to satisfy in the real world.

The nested method solves the MBOP directly by performing the lower level optimization independently for each upper level solution. For example, Deb *et al.* used elite nondominated ranking GA or NSGA-II for both upper and lower level optimization[10]. This algorithm was subsequently upgraded by Sinha *et al.*[22]. Their improved version evaluates the upper level only once, which significantly reduces the number of evaluations of the algorithm. Second, it also allows the file members to participate in crossover, which improves the performance of the algorithm. Deb *et al.* proposed a hybrid EA combined with a local search strategy for optimization[23]. Cai *et al.* proposed a divide-and-conquer strategy, in which all variables (both upper and lower levels) are divided into multiple mutually exclusive groups and optimized separately[20]. Although these methods are successful, due to the structural characteristics of bilevel optimization, they cannot afford the large number of evaluations required to be consumed by the lower level evaluation when the number of upper level solutions is large. In order to solve the more complex MBOP, many new algorithms are produced, and a genetic algorithm is adopted at both levels. For example, both the upper and lower levels use Particle swarm optimization, and both upper and lower levels use differential evolution (DE).

The surrogate-model-based methods use a surrogate model to approximate the constraints and objective function at the lower level, with the aim of reducing the number of lower level fitness evaluations (FEs). For example, Sinha *et al.* proposed an approximation-set-mapping approach which used quadratic functions to address the lower level optimization problem[18]. Sinha *et al.* modeled the lower level decision variables using a value function named m-BLEAQ; they used a quadratic function to estimate unknown lower level decision variables[19]. It is essential to note that the accuracy of the surrogate model has a significant impact on the performance of final solutions.

In the real world, we are often faced with a complex set of problems, which often require multiple goals to be considered at the same time, and each goal may have a mutually constrained relationship. In order to solve this kind of problem, the bilevel optimization algorithm provides an effective solution. The bilevel optimization algorithm decomposes the problem into two levels, the upper optimization part is responsible for generating a set of feasible solutions, and the lower optimization part further finds the lower optimal solutions that meets the constraints according to this set of feasible solutions. This strategy of decomposing the problem enables the bilevel optimization algorithm to comprehensively consider multiple aspects of the problem, so as to obtain more comprehensive and high-quality solutions. Most of the current MBLOPs are from the perspective of game theory, using the concept of objective function values and Pareto optimality. In contrast, Gu *et al.* are the first to study the convergence characteristics of MBLOPs problems from the perspective of traditional optimization, and consider a minimum and maximum robust version of the multi-objective problem, weighing the optimality of different objectives, and ensuring that each objective obtains a single optimal solution, rather than generating multiple Pareto optimal solutions[24]. Recently, Wang *et al.* designed a lower level environment selection strategy and an upper level solution regeneration strategy to improve the search efficiency of the algorithm[25]. Inspired by the biological classification of species, Mejía-de-Dios *et al.* proposed a novel evolutionary framework based on the concept of family[26]. They adopt the concept of science in biology to promote the diversity of solutions.

However, the implementation and application of the bilevel optimization algorithm still face many challenges. First, the complexity of bilevel optimization problems tends to be high, making the solution process very difficult. Secondly, the existing bilevel optimization algorithms are often difficult to balance the relationship between solution speed and solution quality. In order to solve these problems, we need to research and develop new bilevel optimization algorithms to improve the speed and quality of solutions.

In bilevel optimization, the lower level optimization tasks are usually relatively similar for multiple close upper

level solutions. Inspired by this, based on the above observations, we design a new algorithm called CCBMO which utilizes match and cluster. To our knowledge, this is the first time that bilevel optimization has been performed using the similarity of parent and child lower level optimization tasks. The main contributions of this article are summarized as follows:

1. Initially, in the upper level decision space, we match a parent to each offspring based on the Euclidean distance. This makes it easier to select a partial solution from the parent's lower solution as the initial solution when performing a lower optimization for a child, greatly reducing the number of lower level optimizations required.
2. Then, the k-means is used to cluster the upper level offspring into $N_u/2$ groups. When solutions in the same group are being optimized at the next level, we use co-evolution.
3. Following this, environment selection is performed based on upper level objective values, and selected upper level solutions are chosen for iteration.

The remainder of this paper is structured as follows: Section II presents the proposed bilevel optimization algorithm. Section III provides a comprehensive experimental study, analysing and discussing the results. Finally, Section IV summarizes the findings and the potential for further research.

2. THE PROPOSED ALGORITHM

2.1. General framework of proposed algorithm

The framework of the CCBMO algorithm is established by algorithm 1 . Initially, a set of N_u upper level solutions $\mathcal{X}^u = \left\{\mathbf{x}_1^u, \mathbf{x}_2^u, ..., \mathbf{x}_{N_u}^u\right\}$ is randomly generated. Subsequently, the lower level optimization is conducted for $\mathbf{x}_i^u$ individually via algorithm 2. The obtained lower level solutions and their corresponding upper level solutions are stored in $\mathcal{L}^p$ (line 2). The solutions in $\mathcal{L}^p$ are then ranked based on the upper level objective values and the degree of constraint violation, with the nondominated solutions being stored in $\mathcal{X}^o$ (line 3).

During the main loop, we execute the DE on $\mathcal{X}^u$, with the resulting offspring stored in $\mathcal{X}^{uo} = \left\{\mathbf{x}_1^{uo}, ..., \mathbf{x}_{N_u}^{uo}\right\}$ (line 5). For each solution in $\mathcal{X}^u$, the closest solution to $\mathbf{x}_i^{uo}$ is denoted as $\mathbf{x}_i^{um}$ (lines 6-8). Next, we explain the several types of matches that can occur as a result. As shown in Figure 1 , in the first case, $\mathbf{x}_1^{uo}$ is matched with its closest parent upper level solution, $\mathbf{x}_1^u$; therefore, $\mathbf{x}_1^{um} = \mathbf{x}_1^u$. In the second case, $\mathbf{x}_2^{uo}$ and $\mathbf{x}_5^{uo}$ are both matched with $\mathbf{x}_2^u$, as they are the closest to it; therefore, $\mathbf{x}_2^{um} = \mathbf{x}_2^u$ and $\mathbf{x}_5^{um} = \mathbf{x}_2^u$; in the third case, no offspring are closest to the solutions of $\mathbf{x}_6^u$ and $\mathbf{x}_7^u$.

Then $\mathcal{X}^{uo}$ is clustered into $N_u/2$ groups using the k-means, denoted as $\mathcal{G}_1, \ldots, \mathcal{G}_{N_u/2}$ (line 9). K-means is a clustering method based on Euclidean distance, which aims to minimize the sum of the distances of N objects from the nearest center point, and to divide these objects into K groups according to the size of the K value, and to consider the individuals in the same group to be similar. The index of each solution in ith group in $\mathcal{X}^{uo}$ is stored in $\mathcal{G}_i$. Following the execution of the k-means clustering, the solutions belonging to the same cluster exhibit proximity, suggesting similarities among the corresponding tasks of lower level optimization. Therefore, the lower level optimization is performed simultaneously on the solutions belonging to the same group based on algorithm 2. After that, the optimized lower level solutions and their corresponding upper level solutions are subsequently stored in $\mathcal{L}^s$ (lines 11-14). The parents in $\mathcal{L}^p$ and offspring in $\mathcal{L}^s$ are combined, denoted as $\mathcal{L}$. Then, N_u distinct upper level solutions are chosen to replace the original solutions in $\mathcal{X}^u$, based on the nondominated sorting and crowding distance obtained from the upper level environmental selection. The selected upper level solutions and their corresponding lower level solutions are stored in $\mathcal{L}^p$. Finally, the nondominated solutions in $\mathcal{L}$ are selected and merged into $\mathcal{X}^o$ to update it.

In order to select the better solution for each iteration, the selection operator of NSGA-II is applied to $\mathcal{L}$ to obtain $\mathcal{X}^u$, and $\mathcal{X}^u$ is used to store N_u distinct upper level solutions. This selection is based on the upper level

Algorithm 1: Proposed algorithm

1 $\mathcal{X}^u = \left\{\mathbf{x}_1^u, \mathbf{x}_2^u, ..., \mathbf{x}_{N_u}^u\right\}$ ← Generate N_u upper level solutions randomly;
2 $\mathcal{L}^p$ ← Perform lower level optimization for each solution in $\mathcal{X}^u$, and obtain the lower level solutions respectively;
3 $\mathcal{X}^o$ ← Select nondominated solutions from $\mathcal{L}^p$;
4 **while** *Not Termination* **do**
5 $\quad \mathcal{X}^{uo} = \left\{\mathbf{x}_1^{uo}, ..., \mathbf{x}_{N_u}^{uo}\right\}$ ← Perform differential evolution on the solutions in $\mathcal{X}^u$;
6 $\quad$ **for** $i = 1 : N_u$ **do**
7 $\quad\quad \mathbf{x}_i^{um}$ ← Find a solution in $\mathcal{X}^u$ that is the closest to $\mathbf{x}_i^{uo}$;
8 $\quad$ **end**
9 $\quad \left[\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_{N_u/2}\right]$ ← Cluster the solutions in $\mathcal{X}^{uo}$ into $N_u/2$ groups utilizing K-means;
10 $\quad \mathcal{L}^s = \emptyset$;
11 $\quad$ **for** $i = 1 : N_u/2$ **do**
12 $\quad\quad \mathcal{L}_1^s, ..., \mathcal{L}_{|\mathcal{G}_i|}^s$ ← Perform lower level optimization for solutions in the $\mathcal{G}_i$ group;
13 $\quad\quad \mathcal{L}^s \leftarrow \mathcal{L}^s \cup \mathcal{L}_1^s, ..., \mathcal{L}_{|\mathcal{G}_i|}^s$;
14 $\quad$ **end**
15 $\quad \mathcal{L} = \mathcal{L}^p \cup \mathcal{L}^s$;
16 $\quad \mathcal{X}^u$ ← Choose N_u distinct upper level solutions from $\mathcal{L}$ based on the results of upper level environment selection;
17 $\quad \mathcal{L}^p = \left\{\mathcal{L}_1^p, \mathcal{L}_2^p, ..., \mathcal{L}_{N_u}^p\right\}$ ← Store the upper level solutions and their corresponding lower level solutions in $\mathcal{X}^u$;
18 $\quad \mathcal{X}^o$ ← Select the nondominated solutions from $\mathcal{L}$ and merge them with $\mathcal{X}^o$;
19 **end**

objective values and the upper level constraint violation. To be specific, the solutions in $\mathcal{X}^u$ are initially divided into different nondominated sets using the constrained-domination principle. According to this principle, if we have two solutions, $\mathbf{x}_1$ and $\mathbf{x}_2$, $\mathbf{x}_1$ is considered better than $\mathbf{x}_2$ if any of the following conditions are met:

1. $cv(\mathbf{x}_1) = 0$ and $cv(\mathbf{x}_2) = 0$, and $\mathbf{x}_1$ Pareto dominates $\mathbf{x}_2$;
2. $cv(\mathbf{x}_1) = 0$ and $cv(\mathbf{x}_2) > 0$;
3. $cv(\mathbf{x}_1) > 0$ and $cv(\mathbf{x}_2) > 0$, and $cv(\mathbf{x}_1) < cv(\mathbf{x}_2)$.

Then, the best half of the upper level solutions are added to $\mathcal{X}^u$.

2.2. Lower level optimization

Algorithm 2 provides the process of lower level optimization. First, an empty set $\mathcal{S}$ is created (line 1). Then we use k to denote the index of the jth element in $\mathcal{G}_i$. Since we match a solution $\mathbf{x}_i^{um}$ for each $\mathbf{x}_i^{uo}$ (lines 6-8 of algorithm 1), we can find $\mathbf{x}_k^{um}$ corresponding to $\mathbf{x}_k^{uo}$. Then we store the lower level solution corresponding to $\mathbf{x}_k^{um}$ into S. As shown in Figure 2, the $\mathbf{x}_1^{uo}, ..., \mathbf{x}_8^{uo}$ were divided into four groups using the k-means algorithm. For example, $\mathbf{x}_1^{uo}$ is in a separate group. Based on Figure 1, $\mathbf{x}_1^{uo}$ is matched with $\mathbf{x}_1^u$, and the lower level solutions whose upper level solution is $\mathbf{x}_1^u$ are stored in S. $\mathbf{x}_2^{uo}$ and $\mathbf{x}_3^{uo}$ belong to the same group, but they are matched with solutions $\mathbf{x}_2^u$ and $\mathbf{x}_3^u$, respectively. Therefore, the lower level solutions whose upper level solutions are $\mathbf{x}_2^u$ and $\mathbf{x}_3^u$ are placed in S.

After removing the repeat values in S, $\min\{N_l/2, |S|\}$ solutions are randomly selected from S and stored in $\mathcal{X}_1^l$. Additionally, $N_l - \min\{N_l/2, |S|\}$ lower level solutions are randomly generated and stored in $\mathcal{X}_2^l$. Then, $\mathcal{X}_1^l$ and $\mathcal{X}_2^l$ are combined to generate $\mathcal{X}^l$ (lines 7-10 of algorithm 2). During the main loop of algorithm 2, we execute

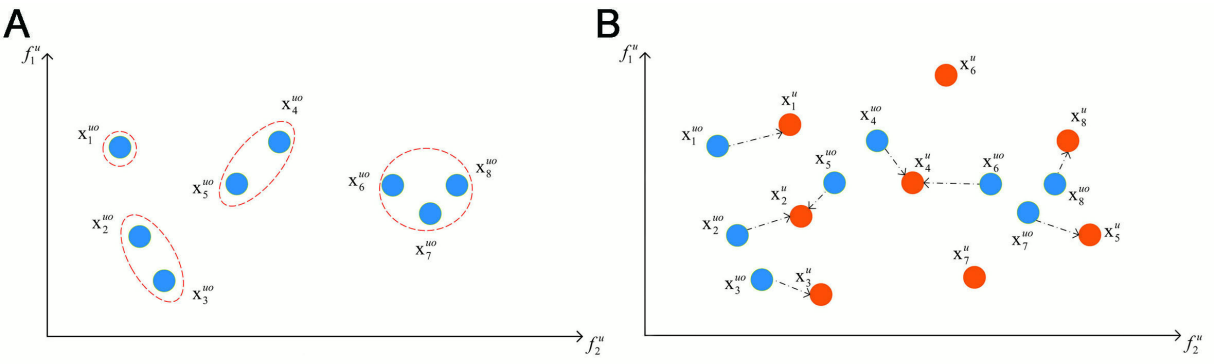

Figure 1. We provide an example of a block plot for matching and clustering in Figure 1A and B. (A) Clustering of $\mathcal{X}^{uo}$ via kmeans; (B) match the solutions in $\mathcal{X}^{u}$ with each solution in $\mathcal{X}^{uo}$ based on the Euclidean distance.

the differential evolution on $\mathcal{X}^l$, with the resulting offspring stored in $\mathcal{X}^{lo}$ (line 12). We merge $\mathcal{X}^l$ and $\mathcal{X}^{lo}$, and store the combined set in C. For solutions in $\mathcal{G}_i$, we choose N_l optimal lower level solutions from C considering the environmental selection of the upper level, and then store them in $\mathcal{A}_1, ..., \mathcal{A}_{|\mathcal{G}_i|}$. Eventually, the lower level solutions merge, replacing the original $\mathcal{X}^l$ (lines 12-18). Upon meeting the termination conditions of the lower level optimization, we store $\mathcal{A}_1, ..., \mathcal{A}_{|\mathcal{G}_i|}$ and their corresponding upper level solutions in $\mathcal{L}^s_1, ..., \mathcal{L}^s_{|\mathcal{G}_i|}$.

Algorithm 2: Lower level optimization

Input: $\mathcal{G}_i$, $\mathcal{L}^p$, $\mathcal{X}^{uo}$;
Output: $\mathcal{L}^s_1, ..., \mathcal{L}^s_{|\mathcal{G}_i|}$;

1 $\mathcal{S} = \emptyset$;
2 **for** $j = 1 : |\mathcal{G}_i|$ **do**
3 $k \leftarrow$ The index of the jth element in $\mathcal{G}_i$;
4 $\mathcal{S}_j \leftarrow$ Find the lower level solutions from $\mathcal{L}^p$ corresponding to the solution whose upper level solution is $\mathbf{x}^{um}_k$; // $\mathbf{x}^{um}_k \in \mathcal{X}^{uo}$;
5 $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_j$;
6 **end**
7 $\mathcal{S} \leftarrow$ Remove repeat lower level solutions from $\mathcal{S}$;
8 $\mathcal{X}^l_1 \leftarrow$ Select min $\{N_l/2, |S|\}$ solutions from $\mathcal{S}$;
9 $\mathcal{X}^l_2 \leftarrow$ Generate $N_l -$ min $\{N_l/2, |S|\}$ lower level solutions randomly;
10 $\mathcal{X}^l \leftarrow \mathcal{X}^l_1 \cup \mathcal{X}^l_2$.
11 **while** *Not termination* **do**
12 $\mathcal{X}^{lo} \leftarrow$ Perform differential evolution on the solutions in $\mathcal{X}^l$;
13 $C \leftarrow \mathcal{X}^l \cup \mathcal{X}^{lo}$;
14 **for** $j = 1 : |\mathcal{G}_i|$ **do**
15 $\mathcal{A}_j \leftarrow$ Perform environmental selection for C;
16 **end**
17 $\mathcal{X}^l \leftarrow \mathcal{A}_1 \cup \mathcal{A}_2 \cup ...\mathcal{A}_{|\mathcal{G}_i|}$;
18 **end**
19 $\mathcal{L}^s_1, ..., \mathcal{L}^s_{|\mathcal{G}_i|} \leftarrow$ Store $\mathcal{A}_1, ..., \mathcal{A}_{|\mathcal{G}_i|}$ and their upper level solutions;

2.3. Termination criterion

Termination conditions for both upper and lower layers are based on their hypervolume (HV). The rate of convergence of HV for both the upper and lower levels is called the H-metric and is calculated as follows:

where $HV^{\max}$ and $HV^{\min}$ denote the maximum and minimum values of the hypervolume in the output solutions, respectively, and $'u'$, $'l'$ are used to distinguish the upper and lower levels respectively. We use the maximum objective value of the nondominated solution as the reference point to calculate the HV.

$$H_u = \frac{HV_u^{\max} - HV_u^{\min}}{HV_u^{\max} + HV_u^{\min}}, H_l = \frac{HV_l^{\max} - HV_l^{\min}}{HV_l^{\max} + HV_l^{\min}} \tag{2}$$

When the number of iterations exceeds σ, and $H < \varepsilon$, the algorithm terminates. We set the termination conditions as $\varepsilon_l = 0.001$; $\varepsilon_u = 0.001$; $\sigma_l = 10$, and $\sigma_u = 40$ dollars. In addition, for upper level optimization, if σ_u exceeds 40 and there are six consecutive generations where H_u remains constant, the algorithm terminates. Since we optimize the lower level solutions simultaneously for multiple upper level solutions, for all lower level optimization processes, if σ_l exceeds 10 and there are five consecutive generations where the value of H_l remains unchanged, the lower level optimization terminates.

3. EXPERIMENTAL STUDIES

In this section, the experimental study for investigating the performance of the proposed algorithm is presented. First, we introduce the test problems and parameter settings. Then we briefly describe the comparison algorithm. Afterward, we introduce the performance metrics used to evaluate the algorithm's performance. Finally, we give the experimental results and analyze the results.

3.1.Test problems

Two benchmark test sets, TP and DS[23], were selected for the experimental studies. In the TP test suite, we chose two test problems, TP1 and TP2, and the DS test suite includes five test problems (denoted as DS1-DS5).

The detailed settings for the TP test set and DS test set are provided in Table 1. D^u and D^l represent the dimensions of the upper and lower level decision variables, respectively. The dimensionality of both upper and lower level decision variables is K = 5 for DS1–DS3. For DS4-DS5, the dimensionality of the upper level decision variables is K = 1, and lower level decision variables is K + L = 10. The other parameters of the comparison algorithms remain the same as in their original papers.

3.2.Compared algorithms and parameter settings

In order to evaluate the performance of the algorithms, two algorithms were chosen for comparison: BLMOCC[20], MOBEA-DPL[27].

(1) BLMOCC: it is an algorithm that uses a knowledge-based variable decomposition strategy to solve a MBOP. The knowledge-based variable decomposition strategy is used throughout the optimization process, and the variables are divided into three groups according to the correlation between the variables in the two levels. Different optimization methods are used independently for different groups.

(2) MOBEA-DPL: it is an algorithm developed on the framework of the nested bilevel multi-objective optimization algorithm. It uses a dual-population optimization strategy to improve the solution of the lower level optimization problem. The first group is used to store nondominated solutions in the lower level, and the second group is used to store upper level solutions that are not dominated by solutions in the first group. Ad-

Table 1. Settings of the test problems

Test problem	N^u	N^l	K	L
TP1	20	20		
TP2	20	20		
DS1	20	20	5	5
DS2	20	20	5	5
DS3	20	20	5	5
DS4	5	40	1	9
DS5	5	40	1	9

Table 2. Performance comparison between BLMOCC, MOBEA-DPL and CCBMO regarding the average values of FEs on TP and DS

Problem	$N_u + N_l$	BLMOCC			MOBEA-DPL			CCBMO		
		FE^u	FE^l	$FE^l + FE^l$	FE^u	FE^l	$FE^l + FE^l$	FE^u	FE^l	$FE^l + FE^l$
TP1	20 + 20	2,536	320,281	322,818	53,499	768,080	821,580	40,615	250,271	290,885
TP2	20 + 20	38,736	1,128,305	1,167,041	53,499	768,080	821,580	36,907	230,748	267,654
DS1 (5 + 5)	20 + 20	30,553	561,769	592,322	50,913	634,383	685,295	36,763	278,767	315,530
DS2 (5 + 5)	20 + 20	36,466	650,363	686,829	55,212	297,359	352,571	66,986	291,608	358,594
DS3 (5 + 5)	20 + 20	4,121	657,513	661,633	65,183	1,243,679	1,308,862	33,257	298,037	331,293
DS4 (1 + 9)	5 + 40	19,492	490,870	510,362	460,000	2,070,000	2,530,000	23,391	99,774	123,165
DS5 (1 + 9)	5 + 40	14,339	362,115	376,454	352,000	1,590,000	1,942,000	56,271	99,959	156,230

FEs: Fitness evaluations.

ditionally, to increase the effectiveness of the search, the offspring of the upper level solutions are selected from the neighborhood of the best solutions in the existing solutions.

3.3.Performance metrics

We limit the maximum FEs of the upper and lower levels to facilitate comparing the number of true evaluations of different algorithms. The maximum number of evaluations for the upper and lower levels is 50,000 and 1,000,000, respectively. FE^u and FE^l represent the number of evaluations in the upper and lower levels, respectively. In addition, we compare the sum of the actual number of evaluations in the upper and lower levels (i.e., $FE^u + FE^l$).

Without loss of generality, we consider using the inverted generational distance (IGD)[28] and hypervolume (HV)[29] to measure the performance of the algorithms. Both metrics can measure the diversity and convergence of the obtained solutions. A smaller IGD value means better algorithm performance, while the opposite is true for HV values, where a larger HV value means better algorithm performance.

3.4.Comparison between CCBMO and other algorithms on each index

In this case, we measure whether the algorithm can converge well with a small number of FEs. We recorded the average of FEs (including the upper level, lower level, and the sum of both levels), the IGD, and HV for all test problems. Specifically, each algorithm ran independently 21 times.

From Table 2, it can be seen that CCBMO has better results than BLMOCC and MOBEA-DPL in terms of lower level average FEs on TP1-TP2, DS1, and DS3-DS5, while slightly worse than BLMOCC in terms of upper level average FEs. This is because in lower level optimization, we introduce the idea of co-optimization, which optimizes similar lower level optimization problems at the same time, thus greatly reducing the number of optimizations and evaluations at the lower level level. However, as shown in Figure 3, due to the obvious advantage of CCBMO in the average FEs of the lower level, the sum of the average FEs of the upper and lower levels shows the best results, except that MOBEA-DPL outperformed CCBMO and BLMOCC on DS2. Obviously, this is all due to the improvement of the lower level optimization strategy.

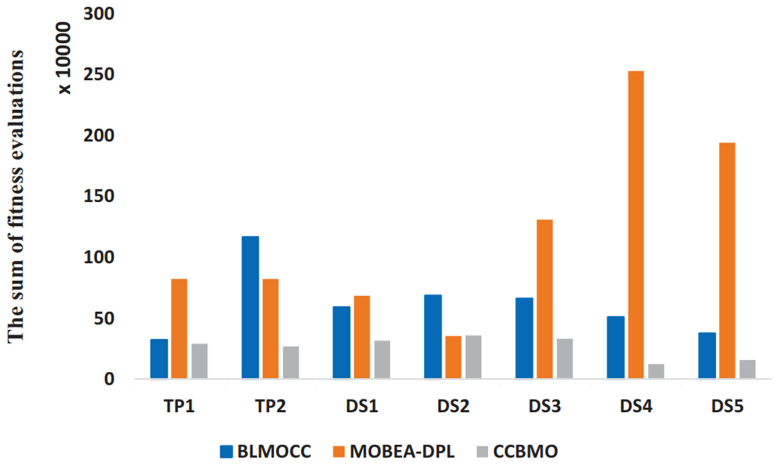

Figure 2. Comparison of the average FEs statistical results of the three algorithms on TP1-TP2 and DS1-DS5. FEs: Fitness evaluations.

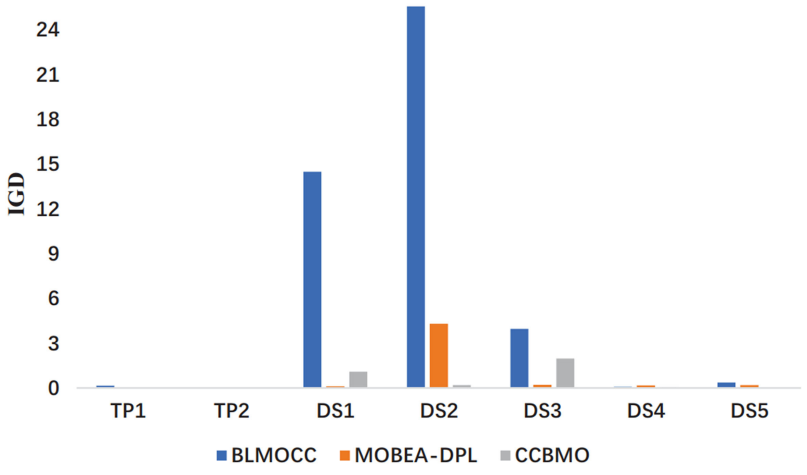

Figure 3. Comparison of the statistical results of the average IGD values of the three algorithms on TP1-TP2 and DS1-DS5. IGD: Inverted generational distance

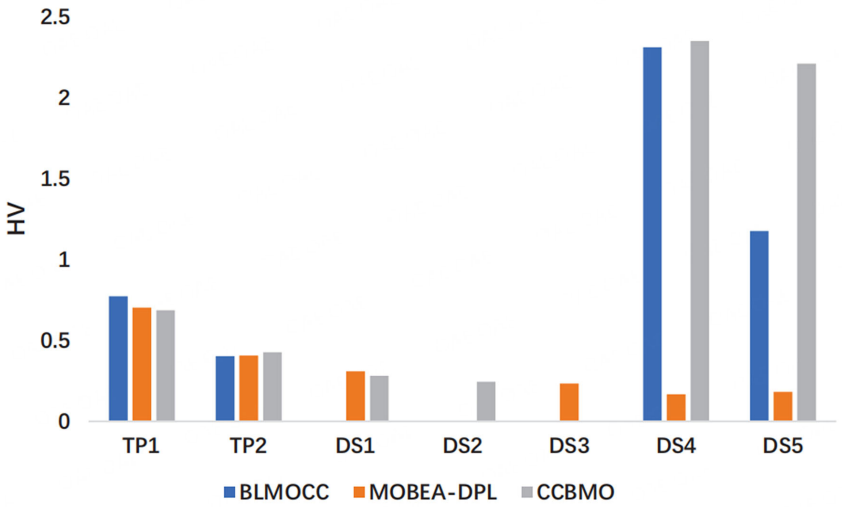

Figure 4. Comparison of the statistical results of the average HV value of the three algorithms on TP1-TP2 and DS1-DS5. HV: Hypervolume.

Table 3. Performance comparison between BLMOCC, MOBEA-DPL and CCBMO regarding the IGD values on TP1-TP2 and DS1-DS5

Problem	$N_u + N_l$	BLMOCC	MOBEA-DPL	CCBMO
TP1	20 + 20	0.1426	0.0126	0.0128
TP2	20+20	0.0310	0.0235	0.0178
DS1 (5 + 5)	20 + 20	14.4842	0.1123	1.0858
DS2 (5 + 5)	20 + 20	25.5732	4.3112	0.1854
DS3 (5 + 5)	20 + 20	3.9562	0.2058	1.9772
DS4 (1 + 9)	5 + 40	0.0768	0.1680	0.0607
DS5 (1 + 9)	5 + 40	0.3582	0.1820	0.0326

IGD: Inverted generational distance.

Table 4. Performance comparison between BLMOCC, MOBEA-DPL and CCBMO regarding the HV values on TP1-TP2 and DS1-DS5

Problem	$N_u + N_l$	BLMOCC	MOBEA-DPL	CCBMO
TP1	20 + 20	0.7723	0.7026	0.6856
TP2	20 + 20	0.4029	0.4074	0.4277
DS1 (5 + 5)	20 + 20	0.0000	0.3092	0.2813
DS2 (5 + 5)	20 + 20	0.0000	0.0028	0.2443
DS3 (5 + 5)	20 + 20	0.0000	0.2342	0.0000
DS4 (1 + 9)	5 + 40	2.3110	0.1680	2.3495
DS5 (1 + 9)	5 + 40	1.1756	0.1820	2.2105

HV: Hypervolume.

Tables 3 and 4 provide the comparative results of three algorithms on seven test problems in terms of IGD and HV. Figures 3 and 4 clearly demonstrate that CCBMO outperformed both BLMOCC and MOBEA-DPL. Next we analyze these experimental results in detail.

Both TP1 and TP2 are non-scalability problems. MOBEA-DPL outperformed other algorithms in terms of IGD value for TP1 due to its lower level dual-population strategy, but it inevitably increased the lower level FEs. CCBMO demonstrates a slightly inferior performance in terms of IGD value compared to MOBEA-DPL. Among all the compared algorithms, BLMOCC obtained the best HV value but the worst IGD value. Regarding TP2, CCBMO achieved the most satisfactory IGD and HV, followed by MOBEA-DPL. BLMOCC performed the worst.

DS1 is often used to evaluate algorithm performance and the ability of the algorithm to coordinate the processing of upper level and lower level tasks. Regarding DS1, MOBEA-DPL achieved the best IGD and HV. CCBMO performed slightly inferior to MOBEA-DPL in terms of IGD and HV values but significantly outperformed BLMOCC.

DS2 is used to assess the algorithm's capability to conduct extensive searches before converging on the frontier. For DS2, CCBMO significantly outperformed the contrasting algorithms. Compared with BLMOCC and MOBEA-DPL, CCBMO could reduce the IGD value by one to two orders of magnitude, respectively, and increase the HV value by two orders of magnitude. BLMOCC and MOBEA-DPL had difficulty achieving satisfactory results in the DS2 test problem.

DS3 involves discrete variables, and as the number of variables increases, the problem becomes increasingly challenging, making it difficult for traditional algorithms to address. MOBEA-DPL achieved the best IGD and HV on DS3. In contrast, CCBMO performed worse than MOBEA-DPL due to its utilization of a traditional nesting method for addressing this problem. BLMOCC had the worst IGD and HV.

Both DS4 and DS5 evaluate the algorithm's capability to search for the appropriate lower level frontier that corresponds to the upper level frontier. Additionally, DS4 necessitates identifying a specific point in the lower level solution that corresponds to the upper level solution. In the case of DS4, CCBMO achieved superior

Table 5. FEs statistics of CCBMO and its variants on TP1-TP2 and DS1-DS5

Problem	$N_u + N_l$	CCBMO	CCBMO-M	CCBMO-C
TP1	20 + 20	290,885	292,893	334,300
TP2	20 + 20	267,654	290,721	358,603
DS1 (5 + 5)	20 + 20	315,530	312,516	363,501
DS2 (5 + 5)	20 + 20	358,594	367,339	379,283
DS3 (5 + 5)	20 + 20	331,293	319,998	406,935
DS4 (1 + 9)	5 + 40	123,165	126,272	147,857
DS5 (1 + 9)	5 + 40	156,230	161,888	119,948

FEs: Fitness evaluations.

Table 6. IGD statistics of CCBMO and its variants on TP1-TP2 and DS1-DS5

Problem	$N_u + N_l$	CCBMO	CCBMO-M	CCBMO-C
TP1	20 + 20	0.0128	0.0156	0.0192
TP2	20 + 20	0.0178	0.1199	0.6835
DS1 (5 + 5)	20 + 20	1.0858	1.0070	3.7125
DS2 (5 + 5)	20 + 20	0.1854	0.2088	0.3523
DS3 (5 + 5)	20 + 20	1.9772	2.1425	2.9403
DS4 (1 + 9)	5 + 40	0.0607	0.0684	0.0635
DS5 (1 + 9)	5 + 40	0.0326	0.0360	0.0571

IGD: Inverted generational distance.

IGD and HV values, which are one and two orders of magnitude better than the corresponding IGD values of MOBEA-DPL and BLMOCC, respectively, and one order of magnitude better than the HV value of MOBEA-DPL. In the case of DS5, CCBMO had significant superiority over the comparison algorithms. In terms of IGD value, CCBMO outperformed both comparison algorithms by two orders of magnitude. The HV value of CCBMO is an order of magnitude better than that of MOBEA-DPL. Therefore, these experiments demonstrate the effectiveness of CCBMO in solving BMOPs.

3.5.Comparison of CCBMO with its variants

To illustrate the effectiveness of the mechanism in CCBMO, we compared two variants of CCBMO, CCBMO-M and CCBMO-C, where CCBMO-M represents the variant that removes the k-means clustering in CCBMO, and CCBMO-C represents the variant that removes the matching between the upper level offspring and parents in CCBMO. Tables 5-7 present the results of CCBMO, CCBMO-M and CCBMO-C regarding the average number of FEs, IGD and HV over 21 runs on TP1-TP2 and DS1-DS5, respectively.

Table 5 gives a comparison of CCBMO and its variants in terms of average FEs. CCBMO consumed the least number of evaluations in solving TP1-TP2, DS2 and DS4. CCBMO-M performed slightly worse than CCBMO, consuming the least number of evaluations on solving DS1 and DS3. CCBMO-C only consumed the least number of evaluations when solving DS5.

Tables 6 and 7 compare the IGD and HV values of CCBMO and its variants (CCBMO-M and CCBMO-C), respectively. In terms of TP1 and DS2-DS5, CCBMO outperformed CCBMO-M and CCBMO-C in both IGD and HV values. For TP2, CCBMO obtained the best IGD value, but the HV value was slightly worse than CCBMO-M. CCBMO-M had the best IGD and HV values when solving DS1. On the other hand, CCBMO-C performed the worst, not obtaining the best IGD and HV values for any problem. Therefore, our experiments demonstrate the effectiveness of matching and clustering incorporated in CCBMO.

Table 7. HV statistics of CCBMO and its variants on TP1-TP2 and DS1-DS5.

Problem	$N_u + N_l$	CCBMO	CCBMO-M	CCBMO-C
TP1	20 + 20	0.6856	0.6298	0.6102
TP2	20 + 20	0.4277	0.4401	0.0239
DS1 (5 + 5)	20+20	0.2813	0.3559	0.0000
DS2 (5 + 5)	20+20	0.2443	0.1919	0.0759
DS3 (5 + 5)	20+20	0.0000	0.0000	0.0000
DS4 (1 + 9)	5+40	2.3495	2.2879	2.3383
DS5 (1 + 9)	5+40	2.2105	2.2082	2.1234

HV: Hypervolume.

4. CONCLUSION

Our work shows the existing bilevel multi-objective optimization algorithms ignore the similarity between the lower level optimization tasks when solving the problem, and thus consume a large number of FEs when performing the lower level optimization. Based on these findings, we have proposed a new optimization algorithm framework (called CCBMO), which utilizes the similarity of the lower level optimization tasks of the parents and offspring to accelerate the solution of the lower level optimization tasks of the offspring. CCBMO mainly includes two stages: a matching stage and a clustering stage. In the matching stage, upper level solutions of the parents and offspring are matched based on Euclidean distance. Each offspring selects the closest parent and inherits its corresponding lower level solutions. The clustering stage subsequently decomposes the offspring into $N_u/2$ groups using k-means. Co-evolution is then performed on the lower level solutions within the same group. CCBMO was tested on seven problems by comparing it with two algorithms (i.e., BLMOCC and MOBEA-DPL). The results demonstrate the effectiveness of CCBMO. At present, the research on bilevel optimization is very limited, and more complex methods are usually used to solve the bilevel optimization problem. In this paper, the traditional genetic algorithm is used to solve the upper level and lower level optimization problems respectively, and the correlation between the upper level and lower level optimization problems is noted, so that the bilevel optimization problem can be solved by a traditional nested method. On the theoretical side, we will continue to study the bilevel optimization problem that solves the NP hard by simple genetic algorithm framework. In addition to this, we will further explore the application of multi-objective bilevel optimization in real life.

Although CCBMO performed well on most problems, we found that CCBMO did not converge well when solving test problems with a large number of decision variables. Therefore, further efforts are needed to develop an efficient selection strategy to address larger-scale BMOPs.

DECLARATIONS

Acknowledgments

Thanks to Yaochu Jin, Xingyi Zhang, Ran Cheng, Ye Tian, Xilu Wang, Dan Guo, Dimo Brockhoff, Handing Wang Qingfu Zhang and Chaoli Sun for providing their source code.

Authors' contributions

Proposal and design of research propositions, drafting or final revision of the paper: Hu WY, Huang PQ
Perform data collection: Hu WY
Technical and material support provided: Li F, Ge EQ

Availability of data and materials

Not applicable.

Financial support and sponsorship

The authors is supported by the National Natural Science Foundation of China under 61903003 and supported by Anhui Provincial Natural Science Foundation (Grant 2308085MF199 and 2008085QE227) and Scientific Research Projects in Colleges and Universities of Anhui Province (Grant 2023AH051124) and Supported by the Open Project of Anhui Province Engineering Laboratory of Intelligent Demolition Equipment (Grant APELIDE2022A007) and Supported by the Open Project of Anhui Province Key Laboratory of Special and Heavy Load Robot (Grant TZJQR001-2021).

Conflicts of interest

All authors declare that they are bound by confidentiality agreements that prevent them from disclosing their conflicts of interest in this work.

Ethical approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

REFERENCES

1. Dempe S, Dinh N, Dutta J, Pandit T. Simple bilevel programming and extensions. *Math Program* 2021;188:227-53. DOI
2. von Stackelberg H. Market structure and equilibrium. Heidelberg: Springer Berlin; 2010. DOI
3. Fortuny-Amat J, McCarl B. A representation and economic interpretation of a two-level programming problem. *J Oper Res Soc* 1981;32:783-92. DOI
4. Dempe S, Kalashnikov V, Pérez-Valdés GA, Kalashnykova N. Bilevel programming problems. Heidelberg: Springer Berlin; 2015. DOI
5. Wogrin S, Pineda S, Tejada-Arango DA. Applications of bilevel optimization in energy and electricity markets. In: Dempe S, Zemkoho A, editors. Bilevel optimization. Cham: Springer; 2020. pp. 139-68. DOI
6. Yin Y. Multiobjective bilevel optimization for transportation planning and management problems. *J Adv Transp* 2002;36:93-105. DOI
7. Liu R, Gao J, Zhang J, Meng D, Lin Z. Investigating bi-level optimization for learning and vision from a unified perspective: a survey and beyond. *IEEE Trans Pattern Anal Mach Intell* 2022;44;10045-67. DOI
8. Eichfelder G. Multiobjective bilevel optimization. *Math Program* 2010;123:419-49. DOI
9. Halter W, Mostaghim S. Bilevel optimization of multi-component chemical systems using particle swarm optimization. In: 2006 IEEE International Conference on Evolutionary Computation; 2006 Jul 16-21; Vancouver, Canada. IEEE; 2006. pp. 1240-47. DOI
10. Deb K, Sinha A. Solving bilevel multi-objective optimization problems using evolutionary algorithms. In: Ehrgott M, Fonseca CM, Gandibleux X, Hao JK, Sevaux M, editors. Lecture notes in computer Science. Heidelberg: Springer Berlin; 2009. pp. 110-24. DOI
11. Jia L, Wang Y. A genetic algorithm for multiobjective bilevel convex optimization problems. In: 2009 International Conference on Computational Intelligence and Security; 2009 Dec 11-14; Beijing, China. IEEE; 2009. pp. 98-102. DOI
12. Li H, Zhang L. An efficient solution strategy for bilevel multiobjective optimization problems using multiobjective evolutionary algorithm. *Soft Comput* 2021;25:8241-61. DOI
13. Jia L, Wang Y. Genetic algorithm based on primal and dual theory for solving multiobjective bilevel linear programming. In: 2011 IEEE Congress of Evolutionary Computation (CEC); 2011 Jun 5-8; New Orleans, LA, USA. IEEE; 2011. pp. 558-65. DOI
14. Li H, Zhang Q, Chen Q, Zhang L, Jiao YC. Multiobjective differential evolution algorithm based on decomposition for a type of multiobjective bilevel programming problems. *Knowl Based Syst* 2016;107:271-88. DOI
15. Pieume CO, Marcotte P, Fotso LP, Siarry P. Generating efficient solutions in bilevel multi-objective programming problems. *Am J Oper Res* 2013;3:289-98. DOI
16. He Q, Lv Y. Particle swarm optimization based on smoothing approach for solving a class of bi-level multiobjective programming problem. *Cybern Inf Technol* 2017;17:59-74. DOI
17. Alves MJ, Dempe S, Júdice JJ. Computing the Pareto frontier of a bi-objective bi-level linear problem using a multiobjective mixed-integer programming algorithm. *Optimization* 2010;61:335-58. DOI
18. Sinha A, Malo P, Deb K. Approximated set-valued mapping approach for handling multiobjective bilevel problems. *Comput Oper Res* 2017;77:194-209. DOI
19. Sinha A, Malo P, Deb K. Towards understanding bilevel multi-objective optimization with deterministic lower level decisions. In: Gaspar-Cunha A, Henggeler Antunes C, Coello C, editors. Lecture notes in computer science. Cham: Springer; 2015. pp. 426-43. DOI

20. Cai X, Sun Q, Li Z, et al. Cooperative coevolution with knowledge-based dynamic variable decomposition for bilevel multiobjective optimization. *IEEE Trans Evol Comput* 2022;26:1553-65. DOI
21. Islam MM, Singh HK, Ray T. A nested differential evolution based algorithm for solving multi-objective bilevel optimization problems. In: Ray T, Sarker R, Li X, editors. Lecture notes in computer science. Cham: Springer; 2016. pp. 101-12. DOI
22. Sinha A, Deb K. Towards understanding evolutionary bilevel multi-objective optimization algorithm. *IFAC Proc Vol* 2009;42:338-43. DOI
23. Deb K, Sinha A. An efficient and accurate solution methodology for bilevel multi-objective programming problems using a hybrid evolutionary-local-search algorithm. *Evol Comput* 2010;18:403-49. DOI
24. Gu A, Lu S, Ram P, Weng TW. Min-max multi-objective bilevel optimization with applications in robust machine learning. arXiv. [Preprint.] Mar 7, 2023 [accessed 2023 Dec 21]. Available from: https://arxiv.org/abs/2203.01924.
25. Wang W, Liu HL. A multi-objective bilevel optimization evolutionary algorithm with local search. In: 2021 17th International Conference on Computational Intelligence and Security (CIS); 2021 Nov 19-22; Chengdu, China. IEEE; 2021. pp. 408-12. DOI
26. Mejía-de-Dios JA, Rodríguez-Molina A, Mezura-Montes E. A novel evolutionary framework based on a family concept for solving multi-objective bilevel optimization problems. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. New York, NY, USA: Association for Computing Machinery; 2022. pp. 348–51. DOI
27. Wang W, Liu HL, Shi H. A multi-objective bilevel optimisation evolutionary algorithm with dual populations lower-level search. *Conn Sci* 2022;34:1556-81. DOI
28. Bosman PAN, Thierens D. The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Trans Evol Comput* 2003;7:174-88. DOI
29. Zitzler E, Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans Evol Comput* 1999;3:257-71. DOI