A Cluster-based Co-evolutionary Optimization Method for Bilevel Multi-objective Optimization

Wanyue Hu¹, Fei Li², Pei-Qiu Huang³, Erqian Ge⁴

¹School of Electrical and Information Engineering, Anhui University of Technology, Maanshan, 243002, China.
²School of Electrical and Information Engineering, Anhui University of Technology, Maanshan, 243002, China.
³School of Automation, Central South University, Changsha 410083, China.
⁴School of Electrical and Information Engineering, Anhui University of Technology, Maanshan, 243002, China.

Correspondence to: No. 1530, Maxiang Road, Ma’anshan City, Anhui Province, 243032, China. E-mail: lanceleeneu@126.com

Received: date month year First Decision: Revised: Accepted: Published:

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

© The Author(s) 2023. Open Access This article is licensed under a Creative Commons Attribution 4.0 International License (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, sharing, adaptation, distribution and reproduction in any medium or format, for any purpose, even commercially, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.
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[1]. 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[2]. Bilevel optimization has its roots in game theory[3]. 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[4–10]. 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:

\[
\begin{align*}
\min_{x^u, x^l} f^u &= \{ f^u_1(x^u, x^l), \ldots, f^u_n(x^u, x^l) \}, \\
\text{s.t.} \quad g^u_j(x^u, x^l) &\leq 0, \quad j = 1, \ldots, p^u, \\
\quad &x^l \in \arg \min_{x^l} f^l = \{ f^l_1(x^u, x^l), \ldots, f^l_m(x^u, x^l) \}, \\
\text{s.t.} \quad g^l_j(x^u, x^l) &\leq 0, \quad j = 1, \ldots, p^l,
\end{align*}
\]

where \(f^u\) and \(f^l\) are the upper level and lower level objective functions, respectively; \(x^u\) and \(x^l\) 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; \(g^u_j(x^u, x^l)\) and \(g^l_j(x^u, x^l)\) are the \(j\)th 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 \(x^l\) is a Pareto-optimal solution to the lower level optimization problem with regard to the given \(x^u\), then a solution \(x = (x^u, x^l)\) 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[11,12] 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[13–18], surrogate-model-based methods[19–21], and nested-based methods[22–25].

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.[13] used adaptive weighting sum scalarization and Karush-Kuhn-Tucker (KKT) conditions to transform a BMOP to a multi-objective optimization problem (MOP). Then they proposed an effective smoothing technique to cope with complementarity constraints. Li et al.[15] improved the algorithm presented in[13]. 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.[14] converted a MBOP into a MOP based on the primal and dual theory and used multi-objective metaheuristics and constraint processing techniques for optimization. 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.[22] used elite nondominated ranking GA or NSGA-II for both upper and lower level optimization. This algorithm was subsequently upgraded by Sinha et al.[25]. 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.[26] proposed a hybrid EA combined with a local search strategy for optimization. Cai et al.[23] 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. 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. [20] proposed an approximation-set-mapping approach which used quadratic functions to address the lower level optimization problem. Sinha et al. [21] 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. 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. [27] 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. Recently, Wang et al. [28] designed a lower level environment selection strategy and an upper-level solution regeneration strategy to improve the search efficiency of the algorithm. Inspired by the biological classification of species, Mezura-Montes et al. [29] proposed a novel evolutionary framework based on the concept of family. 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 \( X^u = \{x_1^u, x_2^u, ..., x_{N_u}^u\} \) is randomly generated. Subsequently, the lower level optimization is conducted for \( x_i^u \) 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 non-dominated solutions being stored in $\mathcal{X}^o$ (line 3).

**Algorithm 1:** Proposed algorithm

1. $\mathcal{X}^u = \{x_1^u, x_2^u, \ldots, x_{N_u}^u\} \leftarrow$ Generate $N_u$ upper level solutions randomly;
2. $\mathcal{L}^P \leftarrow$ Perform lower level optimization for each solution in $\mathcal{X}^u$, and obtain the lower level solutions respectively;
3. $\mathcal{X}^o \leftarrow$ Select non-dominated solutions from $\mathcal{L}^P$;
4. while Not Termination do
   5. $\mathcal{X}^{uo} = \{x_1^{uo}, \ldots, x_{N_u}^{uo}\} \leftarrow$ Perform differential evolution on the solutions in $\mathcal{X}^u$;
   6. for $i = 1 : N_u$ do
      7. $x_i^{um} \leftarrow$ Find a solution in $\mathcal{X}^u$ that is the closest to $x_i^{uo}$;
   8. end
   9. $[G_1, G_2, \ldots, G_{N_u/2}] \leftarrow$ Cluster the solutions in $\mathcal{X}^{uo}$ into $N_u/2$ groups utilizing K-means;
10. $\mathcal{L}^s = \emptyset$;
11. for $i = 1 : N_u/2$ do
12.   $\mathcal{L}_1^s, \ldots, \mathcal{L}_{|G_i|}^s \leftarrow$ Perform lower level optimization for solutions in the $G_i$ group;
13.   $\mathcal{L}^s \leftarrow \mathcal{L}^s \cup \mathcal{L}_1^s, \ldots, \mathcal{L}_{|G_i|}^s$;
14. end
15. $\mathcal{L} = \mathcal{L}^P \cup \mathcal{L}^s$;
16. $\mathcal{X}^u \leftarrow$ Choose $N_u$ distinct upper level solutions from $\mathcal{L}$ based on the results of upper level environment selection;
17. $\mathcal{L}^P = \{L_1^P, L_2^P, \ldots, L_{N_u}^P\} \leftarrow$ Store the upper level solutions and their corresponding lower level solutions in $\mathcal{X}^u$;
18. $\mathcal{X}^o \leftarrow$ Select the non-dominated solutions from $\mathcal{L}$ and merge them with $\mathcal{X}^o$;
19. end

During the main loop, we execute the DE on $\mathcal{X}^u$, with the resulting offspring stored in $\mathcal{X}^{uo} = \{x_1^{uo}, \ldots, x_{N_u}^{uo}\}$ (line 5). For each solution in $\mathcal{X}^u$, the closest solution to $x_i^{uo}$ is denoted as $x_i^{um}$ (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, $x_i^{uo}$ is matched with its closest parent upper level solution, $x_i^u$; therefore, $x_i^{uo} = x_i^u$. In the second case, $x_2^{uo}$ and $x_5^{uo}$ are both matched with $x_6^u$, as they are the closest to it; therefore, $x_2^{uo} = x_2^u$ and $x_5^{uo} = x_5^u$. In the third case, no offspring are closest to the solutions of $x_6^u$ and $x_7^u$.

Then $\mathcal{X}^{uo}$ is clustered into $N_u/2$ groups using the k-means, denoted as $G_1, \ldots, G_{N_u/2}$ (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 $\mathcal{X}^{uo}$ is stored in $G_i$. 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 non-dominated 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 non-dominated 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 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, \( x_1 \) and \( x_2 \), \( x_1 \) is considered better than \( x_2 \) if any of the following conditions are met:

1. \( cv(x_1) = 0 \) and \( cv(x_2) = 0 \), and \( x_1 \) Pareto dominates \( x_2 \);  
2. \( cv(x_1) = 0 \) and \( cv(x_2) > 0 \);  
3. \( cv(x_1) > 0 \) and \( cv(x_2) > 0 \), and \( cv(x_1) < cv(x_2) \).

Then, the best half of the upper level solutions are added to \( X^u \).

### 2.2. Lower Level Optimization

Algorithm 2 provides the process of lower level optimization. First, an empty set \( S \) is created (line 1). Then we use \( k \) to denote the index of the \( j \)th element in \( G_i \). Since we match a solution \( x_{im}^l \) for each \( x_{im}^l \) (lines 6-8 of Algorithm 1), we can find \( x_{im}^l \) corresponding to \( x_{im}^l \). Then we store the lower level solution corresponding to \( x_{im}^l \) into \( S \). As shown in Fig. 2, the \( x_{im}^l \) were divided into four groups using the k-means algorithm. For example, \( x_{im}^l \) is in a separate group. Based on Fig. 1, \( x_{im}^l \) is matched with \( x_{im}^u \), and the lower level solutions whose upper level solution is \( x_{im}^l \) are stored in \( S \). \( x_{im}^l \) and \( x_{im}^u \) belong to the same group, but they are matched with solutions \( x_{im}^l \) and \( x_{im}^l \), respectively. Therefore, the lower level solutions whose upper level solutions are \( x_{im}^l \) and \( x_{im}^l \) are placed in \( S \).

![Figure 1. We provide an example of a block plot for matching and clustering in Figure 1A-B. Left: match the solutions in \( X^u \) with each solution in \( X^l \) based on the Euclidean distance; Right: clustering of \( X^l \) via kmeans.](image)

After removing the repeat values in \( S \), \( \min \{N_i/2, |S|\} \) solutions are randomly selected from \( S \) and stored in \( X^l_i \). Additionally, \( N_i - \min \{N_i/2, |S|\} \) lower level solutions are randomly generated and stored in \( X^l_i \). Then, \( X^l_i \) and \( X^l_j \) are combined to generate \( X^l \) (lines 7-10 of Algorithm 2). During the main loop of Algorithm 2, we execute the differential evolution on \( X^l \), with the resulting offspring stored in \( X^{lo} \) (line 12). We merge \( X^l \) and \( X^{lo} \), and store the combined set in \( C \). For solutions in \( G_i \), we choose \( N_i \) optimal lower level solutions from \( C \) considering the environmental selection of the upper level, and then store them in \( A_1, \ldots, A_{|G_i|} \). Eventually, the lower level solutions merge, replacing the original \( X^l \) (lines 12-18). Upon meeting the termination conditions of the lower level optimization, we store \( A_1, \ldots, A_{|G_i|} \) and their corresponding upper level solutions in \( L^*_1, \ldots, L^*_{|G_i|} \).

#### 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:

\[
H_u = \frac{HV_{max}^u - HV_{min}^u}{HV_{max}^u + HV_{min}^u}, H_l = \frac{HV_{max}^l - HV_{min}^l}{HV_{max}^l + HV_{min}^l}
\]

(2)

When the number of iterations exceeds \( \sigma_u \) 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 \( \sigma_u \) exceeds 40
Algorithm 2: Lower level optimization

Input: $G_i, L^P, X^UO$; 
Output: $L^S_1, ..., L^S_{|G_i|}$;

1 $S = \emptyset$;
2 for $j = 1 : |G_i|$ do
3  $k \leftarrow$ The index of the $j$th element in $G_i$;
4  $S_j \leftarrow$ Find the lower level solutions from $L^P$ corresponding to the solution whose upper level solution is $x^UO_k$; // $x^UO_k \in X^UO$;
5  $S \leftarrow S \cup S_j$;
6 end
7 $S \leftarrow$ Remove repeat lower level solutions from $S$;
8 $X^L_1 \leftarrow$ Select $\min\{N_l/2, |S|\}$ solutions from $S$;
9 $X^L_2 \leftarrow$ Generate $N_l - \min\{N_l/2, |S|\}$ lower level solutions randomly;
10 $X^L \leftarrow X^L_1 \cup X^L_2$.
11 while Not Termination do
12  $X^{UO} \leftarrow$ Perform differential evolution on the solutions in $X^L$;
13  $C \leftarrow X^L \cup X^{UO}$;
14  for $j = 1 : |G_i|$ do
15    $A_j \leftarrow$ Perform environmental selection for $C$;
16  end
17  $X^L \leftarrow A_1 \cup A_2 \cup ... A_{|G_i|}$;
18 end
19 $L^S_1, ..., L^S_{|G_i|} \leftarrow$ Store $A_1, ..., A_{|G_i|}$ and their upper level solutions;

and there are six consecutive generations where $H_2$ 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 $\sigma_l$ 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\cite{26}, 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 $D^I$ 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\cite{23}, MOBEA-DPL\cite{30}.

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. Additionally, 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.

Table 1. SETTINGS OF THE TEST PROBLEMS

<table>
<thead>
<tr>
<th>Test Problem</th>
<th>N_u</th>
<th>N_l</th>
<th>K</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP1</td>
<td>20</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP2</td>
<td>20</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DS1</td>
<td>20</td>
<td>20</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>DS2</td>
<td>20</td>
<td>20</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>DS3</td>
<td>20</td>
<td>20</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>DS4</td>
<td>5</td>
<td>40</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>DS5</td>
<td>5</td>
<td>40</td>
<td>1</td>
<td>9</td>
</tr>
</tbody>
</table>

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. \( F_E^u \) and \( F_E^l \) 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., \( F_E^u + F_E^l \)).

Without loss of generality, we consider using the inverted generational distance (IGD)\(^{[31]}\) and hypervolume (HV)\(^{[32]}\) 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. The best results for each test problem are highlighted in gray.

Table 2. PERFORMANCE COMPARISON BETWEEN BLMOCC, MOBEA-DPL AND CCBMO REGARDING THE AVERAGE VALUES OF FEs ON TP AND DS.

<table>
<thead>
<tr>
<th>Problem</th>
<th>N_u + N_l</th>
<th>BLMOCC</th>
<th>MOBEA-DPL</th>
<th>CCBMO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( F_E^u )</td>
<td>( F_E^l )</td>
<td>( F_E^l + F_E^l )</td>
<td>( F_E^u )</td>
</tr>
<tr>
<td>TP1</td>
<td>20-20</td>
<td>2,936</td>
<td>328,281</td>
<td>332,818</td>
</tr>
<tr>
<td>TP2</td>
<td>20-20</td>
<td>38,736</td>
<td>1,128,305</td>
<td>1,167,041</td>
</tr>
<tr>
<td>DS1(5+5)</td>
<td>20-20</td>
<td>30,553</td>
<td>561,769</td>
<td>592,322</td>
</tr>
<tr>
<td>DS2(5+5)</td>
<td>20-20</td>
<td>36,466</td>
<td>650,363</td>
<td>686,829</td>
</tr>
<tr>
<td>DS3(5+5)</td>
<td>20-20</td>
<td>4,121</td>
<td>657,513</td>
<td>661,633</td>
</tr>
<tr>
<td>DS4(1+9)</td>
<td>5+40</td>
<td>19,492</td>
<td>490,870</td>
<td>510,362</td>
</tr>
<tr>
<td>DS5(1+9)</td>
<td>5+40</td>
<td>14,339</td>
<td>362,115</td>
<td>376,454</td>
</tr>
</tbody>
</table>

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](image-url)

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

![Figure 3](image-url)

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

Table 3 and 4 provide the comparative results of three algorithms on seven test problems in terms of IGD and HV. The figures in Figure 4 and Figure 5 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
Figure 4. Comparison of the statistical results of the average HV value of the three algorithms on TP1-TP2 and DS1-DS5.

Table 3. PERFORMANCE COMPARISON BETWEEN BLMOCC, MOBEA-DPL AND CCBMO REGARDING THE IGD VALUES ON TP1-TP2 AND DS1-DS5.

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

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 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

Table 4. PERFORMANCE COMPARISON BETWEEN BLMOCC, MOBEA-DPL AND CCBMO REGARDING THE HV VALUES ON TP1-TP2 AND DS1-DS5.

<table>
<thead>
<tr>
<th>Problem</th>
<th>( N_u + N_l )</th>
<th>BLMOCC</th>
<th>MOBEA-DPL</th>
<th>CCBMO</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP1</td>
<td>20+20</td>
<td>0.7723</td>
<td>0.7026</td>
<td>0.6856</td>
</tr>
<tr>
<td>TP2</td>
<td>20+20</td>
<td>0.4029</td>
<td>0.4074</td>
<td>0.4277</td>
</tr>
<tr>
<td>DS1(5+5)</td>
<td>20+20</td>
<td>0.0000</td>
<td>0.3092</td>
<td>0.2813</td>
</tr>
<tr>
<td>DS2(5+5)</td>
<td>20+20</td>
<td>0.0000</td>
<td>0.0028</td>
<td>0.2443</td>
</tr>
<tr>
<td>DS3(5+5)</td>
<td>20+20</td>
<td>0.0000</td>
<td>0.2342</td>
<td>0.0000</td>
</tr>
<tr>
<td>DS4(1+9)</td>
<td>5+40</td>
<td>2.3110</td>
<td>0.1680</td>
<td>2.3495</td>
</tr>
<tr>
<td>DS5(1+9)</td>
<td>5+40</td>
<td>1.1756</td>
<td>0.1820</td>
<td>2.2105</td>
</tr>
</tbody>
</table>
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.

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

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

Tables 6 and Tables 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.

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

Authors’ contributions
Proposal and design of research propositions: Hu WY, Huang PQ; Perform data collection: Hu WY; Technical and material support provided: Li F, Ge eq; Drafting or final revision of the paperHu WY, Huang PQ.

Availability of data and materials
Not applicable.

Financial support and sponsorship
This paper was supported in part by the National Natural Science Foundation of China under 61903003, in part by the Open Project of Anhui Province Key Laboratory of Special and Heavy Load Robot under Grant TZJQR001-2021, and in part by the Scientific Research Projects in Colleges and Universities of Anhui Province under Grant KJ2019A0051, and in part by the Nature Science Research Project of Anhui province under Grant NO.2008085QE227, and in part by the Anhui Province Engineering Laboratory of Intelligent Demolition Equipment under Grant APELIDE2022A007.

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.

Copyright
© The Author(s) 2023.
REFERENCES


17. He Q, Lv Y. Particle Swarm Optimization Based on Smoothing Approach for Solving a Class of Bi-Level Multiobjective Programming Problem. *Cybernetics and Information Technologies* 2017 09:17:59–74. DOI


24. Islam MM. A Nested Differential Evolution Based Algorithm for Solving Multi-objective Bilevel Optimization Problems; 2016. . DOI


Available from: [https://doi.org/10.1007/978-3-642-01020-0](https://doi.org/10.1007/978-3-642-01020-0)

dx.doi.org/10.20517/jsegc.xxxx.xx