A Framework for Scalable Bilevel Optimization: Identifying and Utilizing the Interactions between Upper-level and Lower-level Variables

Pei-Qiu Huang and Yong Wang, *Senior Member, IEEE*

***Abstract*—When solving bilevel optimization problems (BOPs) by evolutionary algorithms (EAs), it is necessary to obtain the lower-level optimal solution for each upper-level solution, which gives rise to a large number of lower-level fitness evaluations, especially for large-scale BOPs. It is interesting to note that some upper-level variables may not interact with some lower-level variables. Under this condition, if the value(s) of one/several upper-level variables change(s), we only need to focus on the optimization of the interacting lower-level variables, thus reducing the dimension of the search space and saving the number of lower-level fitness evaluations. This paper proposes a new framework (called GO) to identify and utilize the interactions between upper-level and lower-level variables for scalable BOPs. GO includes two phases: the grouping phase and the optimization phase. In the grouping phase, after identifying the interactions between upper-level and lower-level variables, they are divided into three types of subgroups (denoted as types I-III), which contain only upper-level variables, only lower-level variables, and both upper-level and lower-level variables, respectively. In the optimization phase, if type-I and type-II subgroups only include one variable, a multi-start sequential quadratic programming is designed; otherwise, a single-level EA is applied. In addition, a criterion is proposed to judge whether a type-II subgroup has multiple optima. If multiple optima exist, by incorporating the information of the upper level, we design new objective function and degree of constraint violation to locate the optimistic solution. As for type-III subgroups, they are optimized by a bilevel EA (BLEA). The effectiveness of GO is demonstrated on a set of scalable test problems by applying it to five representative BLEAs. Moreover, GO is applied to the resource pricing in mobile edge computing.**

***Index Terms*—Bilevel optimization, evolutionary algorithms, interaction, upper-level variables, lower-level variables, resource pricing, mobile edge computing**

I. Introduction

A bilevel optimization problem (BOP) is a hierarchical optimization problem, in which a lower-level optimization problem is nested within an upper-level optimization problem [1]. Many practical optimization problems can be modelled as BOPs, such as pricing setting [2], [3], resource allocation [4], [5], optimal control [6], and machine learning [7], [8].

This work was supported in part by the Innovation-Driven Plan in Central South University under Grant 2018CX010, in part by the National Natural Science Foundation of China under Grants 61673397 and 61976225, and in part by the Beijing Advanced Innovation Center for Intelligent Robots and Systems under Grant 2018IRS06. (*Corresponding author: Yong Wang*)

The authors are with the School of Automation, Central South University, Changsha 410083, China, and also with the Mobile Health Ministry of Education-China Mobile Joint Laboratory, Changsha 410008, China. (Email: pqhuang@csu.edu.cn; ywang@csu.edu.cn)

Without loss of generality, a BOP can be formulated as follows:

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

where $\mathbf{x}^u=(x_1^u,\ldots,x_{n^u}^u)$ and $\mathbf{x}^l=(x_1^l,\ldots,x_{n^l}^l)$ are the upper-level and lower-level solutions, respectively; x_i^u and x_i^l are the ith upper-level variable and the ith lower-level variable, respectively; n^u and n^l are the numbers of upper-level and lower-level variables, respectively; $f^u(\mathbf{x}^u,\mathbf{x}^l)$ and $f^l(\mathbf{x}^u,\mathbf{x}^l)$ are the 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 inequality constraint and the jth lower-level inequality constraint, respectively; $h_j^u(\mathbf{x}^u,\mathbf{x}^l)$ and $h_j^l(\mathbf{x}^u,\mathbf{x}^l)$ are the $(j-p^u)$th upper-level equality constraint and the $(j-p^l)$th lower-level equality constraint, respectively; p^u and p^l are the numbers of upper-level and lower-level inequality constraints, respectively; and (q^u-p^u) and (q^l-p^l) are the numbers of upper-level and lower-level equality constraints, respectively.

The degree of constraint violation of $(\mathbf{x}^u,\mathbf{x}^l)$ on the jth upper-level constraint is calculated as follows:

$$cv_j^u(\mathbf{x}^u,\mathbf{x}^l)=\begin{cases}\max\{0,g_j^u(\mathbf{x}^u,\mathbf{x}^l)\}, & j=1,\ldots,p^u\\ \max\{0,|h_j^u(\mathbf{x}^u,\mathbf{x}^l)|-\delta\}, & j=p^u+1,\ldots,q^u\end{cases} \tag{2}$$

where δ is a positive tolerance value (e.g., $\delta=1e-4$) to relax equality constraints. Then, $cv^u(\mathbf{x}^u,\mathbf{x}^l)=\sum_{j=1}^{q^u} cv_j^u(\mathbf{x}^u,\mathbf{x}^l)$ reflects the degree of constraint violation of $(\mathbf{x}^u,\mathbf{x}^l)$ at the upper level. The calculation of the degree of constraint violation at the lower level is similar to that at the upper level in Eq. (2); thus, its expression is omitted.

A solution $\mathbf{x}=(\mathbf{x}^u,\mathbf{x}^l)$ is called a feasible solution of a BOP if and only if it satisfies all constraints at both levels and $\mathbf{x}^l$ is the optimal solution of the lower-level optimization problem corresponding to $\mathbf{x}^u$. [1] The feasible solution with the

[1]From a practical point of view, there is no guarantee that the optimal solution of a complex lower-level optimization problem can be found, such as an NP-hard problem. Therefore, under this condition, it can be reasonably assumed that the near-optimal solution of the lower-level optimization problem is acceptable [27].

TABLE I
EXISTING STUDIES ON BLEAS.

Reference	Category	Solver	Approximation method
[9]	1	GA	N/A
[10]	1	PSO	N/A
[11]	1	Chaos search	N/A
[12]	1	Estimation of distribution algorithm	N/A
[13]	1	GA	N/A
[14]	1	GA	N/A
[15]	2	GA/Linear programming method	N/A
[16]	2	DE/Interior point method	N/A
[17]	2	Ant colony system/Monotonic optimization method	N/A
[18]	2	GA/Variable neighborhood local search algorithm	N/A
[19]	2	PSO/PSO	N/A
[20]	2	PSO/PSO	N/A
[21]	2	DE/DE	N/A
[22]	2	GA/GA	N/A
[23]	2	DE and interior point method/DE and interior point method	N/A
[24]	2	CMA-ES/CMA-ES	N/A
[25]	2	GA/GA	Quadratic approximation
[26]	2	DE/DE	Response surface modes of order 1 and 2 and Kriging
[27]	2	GP/GP	Greedy heuristics

In the third column, the solvers before and after "/" are used in the upper and lower levels, respectively.
In the fourth column, N/A indicates that no approximation method is used.

smallest upper-level objective function value is the optimal solution of a BOP. The goal of solving a BOP is to locate the optimal solution [24].

A. Related Work

Compared with conventional single-level optimization problems (SOPs), the nested structure of BOPs introduces several difficulties to optimization approaches, such as non-linearity, non-convexity, and disconnectedness, even if both the upper-level and lower-level optimization problems are linear programming problems. To date, many classical bilevel optimization approaches have been proposed [28]. They typically transform BOPs to SOPs by taking advantage of Karush-Kuhn-Tucker (KKT) or other optimality conditions, and then solve the transformed SOPs by classical optimization approaches, such as simplex approach [29], branch-and-bound approach [30], descent approach [31], and penalty function approach [32]. These classical bilevel optimization approaches are computationally efficient, but often require strong assumptions on mathematical properties of BOPs, such as linearity, convexity, or smoothness, thus severely limiting their ability to solve complex BOPs.

As a class of population-based heuristic search approaches, evolutionary algorithms (EAs) have been recognized to be promising approaches for dealing with BOPs [33]. A considerable number of bilevel evolutionary algorithms (BLEAs) have been proposed, which can be roughly classified into two categories: 1) single-level reduction-based approaches and 2) nested structure-based approaches. These algorithms are summarized in Table I.

1) Single-level Reduction-based Approaches: This category of approach first reduces BOPs into SOPs and then employs EAs to solve them. For example, Hejazi *et al.* [9] employed the KKT condition of the lower-level optimization problem to transform BOPs into SOPs, and then solved them by genetic algorithm (GA). Subsequently, the same transformation method is combined with various optimization algorithms, such as particle swarm optimization (PSO) [10], chaos search [11], and estimation of distribution algorithm [12]. In addition, Wang *et al.* [13] constructed specific SOPs with two objective functions that are equivalent to the original BOPs. In order to solve the bi-objective SOPs, a new constraint-handling scheme and a specific-design crossover operator are incorporated into GA. Recognizing that the KKT condition contains a set of inequality constraints that are usually difficult to satisfy, Sinha *et al.* [14] proposed a reduction method based on the relaxed KKT condition and then integrated it with GA to solve BOPs. Although this category of approach is able to cope with BOPs involving complex upper-level optimization problems [34], it requires the assumptions on mathematical properties of the lower-level optimization problem for problem transformation.

2) Nested Structure-based Approaches: In this category of approach, the upper-level and lower-level optimization problems are solved in a nested manner. Specifically, an upper-level solution is first generated and its lower-level optimal solution is then obtained. A lot of studies fall into this category [1]. For instance, Mathieu *et al.* [15] applied GA at the upper level and a linear programming method at the lower level. Zhu *et al.* [16] exploited differential evolution (DE) [35] and interior point method for the upper-level and lower-level optimization problems, respectively. Huang *et al.* [17] studied a bilevel optimization approach for cooperative mobile edge computing (MEC), where ant colony system is employed to make the offloading decision at the upper level and the monotonic optimization method is used to allocate the computation resource at the lower level. Chaabani *et al.* [18] employed GA at the upper level and the variable neighborhood local search algorithm at the lower level for bilevel combinatorial optimization problems. It is worth noting that the above studies only adopt EA at the upper level.

For solving more complex BOPs, many methods adopt EAs at both levels. Zhang *et al.* [19] proposed a bilevel

optimization algorithm for the competitive strategic bidding optimization in day-ahead electricity markets. In this algorithm, PSO is applied as the optimizer at each level. Li *et al.* [20] adopted two variants of PSO to solve the upper-level and lower-level optimization problems, respectively. Angelo *et al.* [21] used DE at the upper and lower levels. Subsequently, GA is used in a similar manner in [22]. Islam *et al.* [23] developed a nested bilevel memetic algorithm, where DE and interior point method are performed at both levels. Very recently, He *et al.* [24] designed a bilevel evolution strategy with covariance matrix adaptation, called BL-CMA-ES, which utilizes CMA-ES [36] at each level. In BL-CMA-ES, a sharing mechanism is constructed to extract prior information of the lower-level optimization problem from the upper-level optimization problem. Although the successes of these methods have been reported, most of them investigate instances with no more than (10+10) variables [2]. The reason is that in these methods, the lower-level optimization process is performed for each upper-level solution. As the number of upper-level variables increases, more upper-level solutions are required in the optimization process; therefore, the number of lower-level FEs may increase dramatically [1]. When solving large-scale BOPs [3], they may suffer from an unaffordable number of lower-level FEs.

To improve the efficiency of nested structure-based approaches, some attempts have been made to incorporate approximation models. There are two ways to achieve this goal. The first way is to construct an approximate mapping between the lower-level optimal solution and the upper-level solution. Sinha *et al.* [25], [38] introduced a bilevel optimization algorithm based on the quadratic approximation, called BLEAQ. In BLEAQ, the lower-level optimal solution is generated by the quadratic approximation, if the quadratic fit is successful. Later, an extension of BLEAQ is proposed by incorporating archiving and local search [39]. The second way is to build the surrogate model for approximating the objective function and constraints at the lower level. Afterward, some lower-level solutions are evaluated by the surrogate model rather than the original objective function and constraints. Islam *et al.* [26] studied a multi-surrogate assisted EA for solving BOPs, where response surface modes of order 1 and 2 and Kriging [4] are used to approximate the objective function and constraints of the lower-level optimization problem. Kieffer *et al.* [27] developed a genetic programming (GP) hyper-heuristic approach to train greedy heuristics for tackling the lower-level optimization problem. These methods can alleviate the computational burden to some extent. However, the choice of the approximation model is still an open issue and is highly problem-dependent [41]. In addition, the construction cost and accuracy of the approximation model are often related to the scale of BOPs; thus, solving large-scale BOPs remains a challenging issue.

B. Motivation and Contributions

In nested structure-based approaches, the lower-level solution should be optimized for each upper-level solution, which leads to a large number of lower-level FEs. This is one of the main reasons why it is difficult for current nested structure-based approaches to solve large-scale BOPs. Note, however, that some lower-level variables may not interact with some upper-level variables. Under this condition, the optimization of lower-level variables depends only on some upper-level variables that interact with them. Therefore, if the value(s) of one/several upper-level variables change(s), we only need to focus on the optimization of the interacting lower-level variables and ignore the optimization of the non-interacting lower-level variables.

To illustrate this phenomenon, we took the following BOP as an example:

$$\begin{aligned} \min_{\mathbf{x}^u,\mathbf{x}^l} \ & f^u = (x_1^u - 1)^2 + (x_2^u - x_1^l)^2 - (x_3^u - x_2^l)^2 \\ \text{s.t.} \ & -1 \le x_1^u, x_2^u, x_3^u \le 1 \\ & \mathbf{x}^l \in \arg\min_{\mathbf{x}^l} f^l = (x_2^u - x_1^l)^2 + (x_3^u - x_2^l)^2 + (x_3^l)^2 \\ & \text{s.t.} \ -1 \le x_1^l, x_2^l, x_3^l \le 1 \end{aligned} \tag{3}$$

where $\mathbf{x}^u = (x_1^u, x_2^u, x_3^u)$ is the upper-level solution and $\mathbf{x}^l = (x_1^l, x_2^l, x_3^l)$ is the lower-level solution.

By analyzing this BOP, the relationship between $\mathbf{x}^u = (x_1^u, x_2^u, x_3^u)$ and the lower-level optimal solution $\mathbf{x}^{l^*} = (x_1^{l^*}, x_2^{l^*}, x_3^{l^*})$ can be obtained as follows:

$$\begin{cases} x_1^{l^*} = x_2^u \\ x_2^{l^*} = x_3^u \\ x_3^{l^*} = 0 \end{cases} \tag{4}$$

From Eq. (4), we can identify the following interactions:

- Any lower-level variable does not interact with x_1^u;
- x_1^l only interacts with x_2^u;
- x_2^l only interacts with x_3^u;
- x_3^l does not interact with any upper-level variable.

Therefore, we can divide upper-level and lower-level variables into four subgroups: $\{x_1^u\}$, $\{x_2^u, x_1^l\}$, $\{x_3^u, x_2^l\}$, and $\{x_3^l\}$, which can be optimized separately. In this way, if the value of x_1^u changes, none of lower-level variables need to be optimized; if the value of x_2^u changes, only x_1^l needs to be optimized; and if the value of x_3^u changes, only x_2^l needs to be optimized.

Based on the above observations, we design a brand-new framework, called GO [5], to identify and utilize the interactions between upper-level and lower-level variables for scalable BOPs. In bilevel optimization, the definition of the interactions between upper-level and lower-level variables is: the optimum of a lower-level variable depends on the values taken by other upper-level variables. To the best of our knowledge, it is the

[2]In this paper, $(n^u + n^l)$ variables denote n^u upper-level variables and n^l lower-level variables.

[3]The existing studies have not yet given a clear definition of large-scale BOPs. However, according to the results in Tables S-I–S-V of the supplementary file, the existing BLEAs require more than $3e+6$ FEs on some (25+25)-variable BOPs. Since the number of FEs is usually set to $3e + 6$ in large-scale single-level optimization [37], we define BOPs with no less than (25+25) variables as large-scale BOPs.

[4]Kriging is a commonly used surrogate model, also known as Gaussian process [40].

[5]The name of GO is derived from the first capital of the two phases in the framework, i.e., *g*rouping phase and *o*ptimization phase.

Algorithm 1 GO

1: **/*Grouping phase*/**
2: Group upper-level and lower-level variables into a set of subgroups $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$ by **Algorithm 3**;
3: **/*Optimization phase*/**
4: Optimize $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$ by **Algorithm 4**;
5: Output the optimal solution of the original BOP by combining the best solution of each subgroup

first attempt to take advantage of the interactions between upper-level and lower-level variables for bilevel optimization.

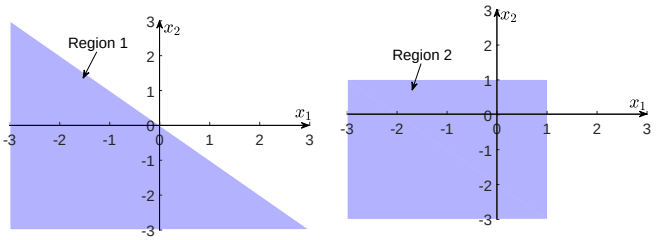
The main contributions of this paper are summarized as follows:

- We first identify the interactions between upper-level and lower-level variables, and then divide them into three types of subgroups (denoted as types I-III). By doing this, a BOP is solved by optimizing several small-scale subgroups, thus reducing the dimension of the search space and saving the number of lower-level FEs. Therefore, GO provides an effective way to tackle large-scale BOPs.
- A multi-start sequential quadratic programming (SQP) is designed to optimize type-I and type-II subgroups that contain only one variable, and a single-level EA is applied to optimize type-I and type-II subgroups that contain multiple variables. In terms of type-III subgroups, they are optimized by a BLEA.
- A criterion is proposed to judge whether a type-II subgroup has multiple optima. If multiple optima exist, new objective function and degree of constraint violation are designed to locate the optimistic solution by considering the information of the upper level.
- Five representative BLEAs, which use different EA paradigms, i.e., GA, DE, and CMA-ES, are embedded into GO. Systematic experiments demonstrate that GO is able to improve the performance of these BLEAs on a set of scalable test problems with up to $(100+100)$ variables and the resource pricing in MEC.

The rest of this paper is organized as follows. Section II describes the details of GO. The experimental studies are presented in Section III. Section IV applies GO to a practical case. Finally, Section V concludes this paper.

II. Proposed Approach

The implementation of GO is presented in **Algorithm 1**, which includes two phases, i.e., the grouping phase and the optimization phase. In the grouping phase, the interactions between upper-level and lower-level variables are first identified by **Algorithm 2**. Afterward, they are divided into a set of subgroups $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$ by **Algorithm 3**. Note that the number of subgroups (i.e., k) is unknown *a priori*. In the optimization phase, $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$ are optimized by **Algorithm 4**. Finally, the optimal solution of the original BOP is output which consists of the best solutions of all subgroups. Overall, GO can be considered as a divide-and-conquer framework. Next, we introduce the grouping phase and the optimization phase in detail.



(a) (b)

Fig. 1. Feasible region of $g(x_1, x_2)$. (a) Original feasible region of $g(x_1, x_2)$, as shown in Region 1. (b) Feasible region of $g(x_1, x_2)$ after x_1 and x_2 are optimized separately, as shown in Region 2.

A. Grouping Phase

As shown in the example in Eq. (3), if we can obtain the relationship between the upper-level solution and the lower-level optimal solution in Eq. (4), it is easy to identify the interactions between upper-level and lower-level variables. However, it is necessary to analyze the mathematical properties of BOPs for building the analytical form of the lower-level optimal solution, which is not a trivial task. The reasons are derived from three aspects:

- The mathematical expressions of the objective function and constraints of black-box BOPs are not available;
- For some complex practical BOPs, it is difficult to carry out problem analysis without professional knowledge;
- For building the analytical form of the lower-level optimal solution, it usually requires strong assumptions on the mathematical properties of BOPs.

Therefore, it is critical to devise an efficient interaction identification method for bilevel optimization, which does not analyze the mathematical properties of BOPs.

For identifying the interaction between two variables, Omidvar *et al.* [42] proposed the following proposition:

Proposition 1. *Two variables x_i and x_j are interacting if there exist* $\mathbf{x}$*, $a_1 \neq a_2$, and $b_1 \neq b_2$ meeting* [6]

$$\Delta f(\mathbf{x})|_{x_j=b_1} \neq \Delta f(\mathbf{x})|_{x_j=b_2} \tag{5}$$

where

$$\begin{aligned}\Delta f(\mathbf{x})|_{x_j=b_1} &= f(\cdots, x_{i-1}, a_2, x_{i+1}, \cdots, x_{j-1}, b_1, x_{j+1}, \cdots) \\ &\quad - f(\cdots, x_{i-1}, a_1, x_{i+1}, \cdots, x_{j-1}, b_1, x_{j+1}, \cdots)\end{aligned}$$

$$\begin{aligned}\Delta f(\mathbf{x})|_{x_j=b_2} &= f(\cdots, x_{i-1}, a_2, x_{i+1}, \cdots, x_{j-1}, b_2, x_{j+1}, \cdots) \\ &\quad - f(\cdots, x_{i-1}, a_1, x_{i+1}, \cdots, x_{j-1}, b_2, x_{j+1}, \cdots)\end{aligned}$$

and $f(\cdot)$ is the objective function.

However, Omidvar *et al.* [42] did not consider the interactions caused by constraints. In fact, if two variables are interacting with respect to constraints, they cannot be optimized separately, even if they are not interacting with respect to the objective function. One may argue that by replacing the

[6] As stated in [43], due to the limited precision of floating-point numbers, this check is not practical on computing devices. Therefore, it is converted to $\left|\Delta f(\mathbf{x})|_{x_j=b_1} - \Delta f(\mathbf{x})|_{x_j=b_2}\right| > \epsilon$, where ϵ is a small value.

Algorithm 2 Identifying the Interaction between Variables x_i and x_j

1: Let $\mathbf{x}^{min} = \{x_1^{min}, \ldots, x_n^{min}\}$ denote the lower bound of the solution and $\mathbf{x}^{max} = \{x_1^{max}, \ldots, x_n^{max}\}$ denote the upper bound of the solution;
2: $a_1 = x_i^{min}$ and $b_1 = x_j^{min}$;
3: $a_2 = x_i^{min} + rand * (x_i^{max} - x_i^{min})$ and $b_2 = x_j^{min} + rand * (x_j^{max} - x_j^{min})$; // $rand$ is a random number uniformly distributed over $(0, 1]$
4: If Eq. (5) in **Proposition** 1 or Eq. (6) in **Proposition** 2 is satisfied, then x_i and x_j are interacting.

objective function with constraints, **Proposition** 1 can be used to identify the interactions caused by constraints. However, it still has some limitations. Taking $g(x_1, x_2) = x_1 + x_2 \leq 0$ as an example, it is clear that x_1 and x_2 are non-interacting according to **Proposition** 1. Thus, $g(x_1, x_2) = x_1 + x_2 \leq 0$ can be decomposed into two subconstraints: $g_1 = x_1 - 1 \leq 0$ and $g_2 = -1 + x_2 \leq 0$, where $g_1 = x_1 - 1 \leq 0$ is generated by fixing the value of x_2 (e.g., let $x_2 = -1$) and $g_2 = -1 + x_2 \leq 0$ is generated by fixing the value of x_1 (e.g., let $x_1 = -1$). However, by doing this, the feasible region of $g(x_1, x_2)$ changes from region 1 in Fig. 1(a) to region 2 in Fig. 1(b), which may result in the change of the optimal solution. Therefore, **Proposition** 1 may fail to identify the interactions caused by constraints.

In this paper, if two variables are involved in the same constraint, they are considered to be interacting. In other words, two variables interact with each other if altering the value of either of them will cause the value of the same constraint to change. Thus, we give the following proposition:

Proposition 2. *Two variables x_i and x_j are interacting if there exist* $\mathbf{x}$*, $a_1 \neq a_2$, and $b_1 \neq b_2$ meeting*

$$\sum_{m=1}^{q} (\left|\Delta g_m(\mathbf{x})|_{x_i=a_1}\right| * \left|\Delta g_m(\mathbf{x})|_{x_j=b_1}\right|) \neq 0 \tag{6}$$

where

$$\begin{aligned}
&\Delta g_m(\mathbf{x})|_{x_i=a_1} \\
=&g_m(\cdots, x_{i-1}, a_1, x_{i+1}, \cdots, x_{j-1}, b_1, x_{j+1}, \cdots) \\
&- g_m(\cdots, x_{i-1}, a_1, x_{i+1}, \cdots, x_{j-1}, b_2, x_{j+1}, \cdots) \\
&\Delta g_m(\mathbf{x})|_{x_j=b_1} \\
=&g_m(\cdots, x_{i-1}, a_1, x_{i+1}, \ldots, x_{j-1}, b_1, x_{j+1}, \cdots) \\
&- g_m(\cdots, x_{i-1}, a_2, x_{i+1}, \cdots, x_{j-1}, b_1, x_{j+1}, \cdots)
\end{aligned}$$

$g_m(\cdot)$ is the mth constraint, and q is the number of constraints.

In **Proposition** 2, Eq. (6) signifies that altering the value of x_i or x_j will result in the change of the value of $g_m(\cdot)$.

To sum up, two variables are interacting if Eq. (5) in **Proposition** 1 or Eq. (6) in **Proposition** 2 is satisfied. Since they are performed based on sampling, **Propositions** 1 and 2 do not depend on the mathematical properties of optimization problems. **Algorithm 2** gives the process of identifying the interaction between two variables, where the settings of a_1, a_2, b_1, and b_2 refer to [42] and [43].

Indeed, the interaction between two variables identified by **Algorithm 2** is called the direct interaction. There also exists

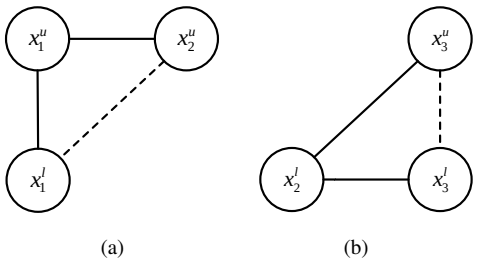


Fig. 2. Direct and indirect interactions between two variables in bilevel optimization, where the solid line indicates the direct interaction and the dashed line indicates the indirect interaction. (a) Indirect interaction caused by two upper-level variables. (b) Indirect interaction caused by two lower-level variables.

Algorithm 3 Variable Grouping

1: $\mathbf{M}_{(n^u+n^l)\cdot(n^u+n^l)} = \mathbf{0}$; //$\mathbf{M}_{(n^u+n^l)\cdot(n^u+n^l)}$ is an interaction matrix with $(n^u + n^l)$ rows and $(n^u + n^l)$ columns
2: $FEs^u = 0$; // FEs^u is the number of upper-level FEs
3: $FEs^l = 0$; // FEs^l is the number of lower-level FEs
4: **for** $i = n^u + 1 : n^u + n^l$ **do**
5: // Identify the interactions between the upper-level and lower-level variables
6: **for** $j = 1 : n^u$ **do**
7: Identify the interaction between $x_{i-n^u}^l$ and x_j^u by using the objective function and constraints at the lower level based on **Algorithm 2**;
8: If $x_{i-n^u}^l$ and x_j^u are interacting, then $M_{ji} = 1$; // M_{ji} denotes the element in the jth row and the ith column of $\mathbf{M}_{(n^u+n^l)\cdot(n^u+n^l)}$
9: **end for**
10: // Identify the interactions among the lower-level variables
11: **for** $j = n^u + 1 : n^u + i - 1$ **do**
12: Identify the interaction between $x_{i-n^u}^l$ and $x_{j-n^u}^l$ by using the objective function and constraints at the lower level based on **Algorithm 2**;
13: If $x_{i-n^u}^l$ and $x_{j-n^u}^l$ are interacting, then $M_{ji} = 1$;
14: **end for**
15: **end for**
16: // Identify the interactions among the upper-level variables
17: **for** $i = 1 : n^u$ **do**
18: **for** $j = 1 : i - 1$ **do**
19: Identify the interaction between x_i^u and x_j^u by using the objective function and constraints at the upper level based on **Algorithm 2**;
20: If x_i^u and x_j^u are interacting, then $M_{ji} = 1$;
21: **end for**
22: **end for**
23: Update FEs^u and FEs^l;
24: Divide the upper-level and lower-level variables into a set of subgroups $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$ based on the maximum connected subgraphs of $\mathbf{M}_{(n^u+n^l)\cdot(n^u+n^l)}$;
25: Categorize $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k$ into three types (types I-III).

the indirect interaction between two variables [44]. Next, we will explain the indirect interaction in bilevel optimization. As shown in Fig. 2(a), both x_2^u and x_1^l interact with x_1^u; therefore, x_2^u and x_1^l are considered to be indirectly interacting. The reason is that the change of the value of x_2^u has an indirect effect on the optimization of x_1^l. This indirect interaction is caused by the interaction between two upper-level variables. Similarly, in Fig. 2(b), x_3^u and x_3^l are considered to be indirectly interacting, which is caused by the interaction between two

Algorithm 4 Optimization of Subgroups

Randomly generate a context vector $\mathbf{c}$ and k initial populations $\{\mathcal{P}_1^0, \mathcal{P}_2^0, \ldots, \mathcal{P}_k^0\}$; // $\mathcal{P}_i^0$ $(i \in \{1, 2, \ldots, k\})$ represents the initial population of $\mathcal{G}_i$
$t = 0$; // t is the generation number;
$\mathbf{s} = [s_1, \ldots, s_k] = \mathbf{0}$; // $s_i = 0$ $(i \in \{1, 2, \ldots, k\})$ indicates that the optimization of $\mathcal{G}_i$ does not terminate; otherwise, $s_i = 1$
$\mathbf{m} = [m_1, \ldots, m_k] = \mathbf{0}$; // $m_i = 1$ $(i \in \{1, 2, \ldots, k\})$ indicates that $\mathcal{G}_i$ is a type-II subgroup and has multiple optima; otherwise, $m_i = 0$
while at least one element in $\mathbf{s}$ is equal to 0 **do**
  for $i = 1 : k$ **do**
    if $s_i == 0$ **then**
      if $\mathcal{G}_i$ is a type-I subgroup **then**
        Implement **Algorithm 5**;
      else if $\mathcal{G}_i$ is a type-II subgroup **then**
        Implement **Algorithm 6**;
      else if $\mathcal{G}_i$ is a type-III subgroup **then**
        Implement a BLEA to produce $\mathcal{P}_i^{t+1}$ and update s_i;
      end if
    end if
  end for
  $t = t + 1$;
end while
Output the best solution of each subgroup

lower-level variables. Overall, the indirect interaction occurs due to the interactions among the upper-level variables or the interactions among the lower-level variables. Therefore, we should implement the following three steps based on **Algorithm 2**:

- Identify the interactions between the upper-level and lower-level variables;
- Identify the interactions among the lower-level variables;
- Identify the interactions among the upper-level variables.

Afterward, according to the maximum connected subgraphs [44], the upper-level and lower-level variables are divided into a set of subgroups: $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$, where k is the number of subgroups. Two variables, which directly and indirectly interact with each other, are in the same subgroup. $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$ can be categorized into the following three types:

- Type-I subgroups, which contain only upper-level variables;
- Type-II subgroups, which contain only lower-level variables;
- Type-III subgroups, which contain both upper-level and lower-level variables.

Algorithm 3 presents the procedure of variable grouping.

B. Optimization Phase

In the optimization phase, we first randomly generate a context vector $\mathbf{c}$ and k initial populations $\{\mathcal{P}_1^0, \mathcal{P}_2^0, \ldots, \mathcal{P}_k^0\}$, where $\mathcal{P}_i^0$ $(i \in \{1, 2, \ldots, k\})$ represents the initial population of $\mathcal{G}_i$. Afterward, the generation number t is set to 0, and both $\mathbf{s} = [s_1, \ldots, s_k]$ and $\mathbf{m} = [m_1, \ldots, m_k]$ are initialized to $\mathbf{0}$. $s_i = 0$ $(i \in \{1, 2, \ldots, k\})$ indicates that the optimization of $\mathcal{G}_i$ does not terminate; otherwise, $s_i = 1$. $m_i = 1$ $(i \in \{1, 2, \ldots, k\})$ indicates that $\mathcal{G}_i$ is a type-II subgroup and has multiple optima; otherwise, $m_i = 0$. Then,

Algorithm 5 Optimization of Type-I Subgroups

if $|\mathcal{G}_i|$ is equal to 1 **then**
  Randomly generate population $\mathcal{S}_i^t$;
  Perform the the multi-start SQP on each individual in $\mathcal{S}_j^t$ to generate an offspring population $\mathcal{O}_i^t$;
else
  Perform the evolutionary operators on $\mathcal{P}_i^t$ to generate $\mathcal{O}_i^t$;
end if
Complement individuals in $\mathcal{O}_i^t$ to be complete solutions by using the context vector $\mathbf{c}$;
Compute the objective function values and degree of constraint violation of $\mathcal{O}_i^t$ at the upper level;
Update FEs^u;
Execute the feasibility rule on the union of $\mathcal{P}_i^t$ and $\mathcal{O}_i^t$ to obtain $\mathcal{P}_i^{t+1}$;
if the maximum number of FEs is reached or the population has converged **then**
  $s_i = 1$;
end if
Return $\mathcal{P}_i^{t+1}$ and s_i

$\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$ are optimized separately in a round-robin fashion until the optimization of all subgroups terminates. **Algorithm 4** describes the optimization of subgroups. In the following, the optimization of type-I, type-II, and type-III subgroups is introduced, respectively.

1) Optimization of Type-I Subgroups: Type-I subgroups are optimized by the single-level optimization algorithm, as they only involve upper-level variables. In addition, since type-I subgroups may only include one variable, using an EA as the optimizer will waste a lot of FEs [45]. In this case, SQP is employed. However, the performance of SQP depends heavily on the start point and an inappropriate start point may lead to performance degradation. To this end, we develop a multi-start SQP, in which a group of start points, rather than one start point, are used. As for type-I subgroups which include multiple variables, we adopt a single-level EA as the optimizer.

The optimization of type-I subgroups is described in **Algorithm 5**. If the number of variables in $\mathcal{G}_i$, denoted as $|\mathcal{G}_i|$, is equal to 1, we first randomly generate population $\mathcal{S}_i^t$ as the start point set, whose size is the same as $\mathcal{P}_i^t$. Then, SQP is performed on each individual in $\mathcal{S}_i^t$ to generate an offspring population $\mathcal{O}_i^t$. In contrast, if $|\mathcal{G}_i|$ is greater than 1, $\mathcal{O}_i^t$ is generated by implementing the evolutionary operators on $\mathcal{P}_i^t$. Afterward, we complement individuals in $\mathcal{O}_i^t$ by using the context vector $\mathbf{c}$ and compute the objective function value and degree of constraint violation of each individual at the upper level. Subsequently, the selection operator is executed on the union of $\mathcal{P}_i^t$ and $\mathcal{O}_i^t$ to obtain $\mathcal{P}_i^{t+1}$ for the next generation based on the commonly used feasibility rule [46]. The optimization of $\mathcal{G}_i$ terminates (i.e., $s_i = 1$) until the stopping criteria are met (i.e., the maximum number of FEs is reached or the population has converged).

2) Optimization of Type-II Subgroups: Like type-I subgroups, type-II subgroups also contain only single-level variables, i.e., lower-level variables. As a result, they are also solved by the the multi-start SQP and a single-level EA. Note, however, that type-II subgroups may have multiple optima, which results in an upper-level solution having multiple diff-

Algorithm 6 Optimization of Type-II Subgroups

1: **if** $|\mathcal{G}_i|$ is equal to 1 **then**
2: Randomly generate population $\mathcal{S}_i^t$;
3: Perform the the multi-start SQP on each individual in $\mathcal{S}_i^t$ to generate an offspring population $\mathcal{O}_i^t$;
4: **else**
5: Perform the evolutionary operators on $\mathcal{P}_i^t$ to generate $\mathcal{O}_i^t$;
6: **end if**
7: Complement individuals in $\mathcal{O}_i^t$ to be complete solutions by using the context vector $\mathbf{c}$;
8: **if** $m_i == 0$ **then**
9: Compute the objective function values and degree of constraint violation of $\mathcal{O}_i^t$ at the lower level;
10: **else**
11: Compute the objective function values and degree of constraint violation of $\mathcal{O}_i^t$ based on Eqs. (10) and (11);
12: **end if**
13: Update FEs^u and FEs^l;
14: Execute the feasibility rule on the union of $\mathcal{P}_i^t$ and $\mathcal{O}_i^t$ to obtain $\mathcal{P}_i^{t+1}$
15: **if** $m_i == 0$ and Eq. (7) is satisfied **then**
16: $m_i = 1$;
17: Recompute the objective function values and degree of constraint violation of $\mathcal{P}_i^{t+1}$ based on Eqs. (10) and (11);
18: **end if**
19: **if** the maximum number of FEs is reached or the population has converged **then**
20: $s_i = 1$;
21: **end if**
22: Return $\mathcal{P}_i^{t+1}$, s_i, and m_i

erent objective function values and/or degree of constraint violation. As in [24], [38] and [47], among these optima, one of the optimal solutions that makes the upper-level optimization problem perform best is selected in this paper, which is called the optimistic solution. However, in order to reach the optimistic solution, there are two issues:

- How to judge whether the subgroup has multiple optima?
- How to locate the optimistic solution if multiple optima exist?

Considering the first issue, we devise a judgment criterion, in which the subgroup is considered to have multiple optima if

$$\text{ave}(\text{std}(\text{norm}_1(\mathcal{P}_i)))) > \theta_1 \wedge \text{std}(\text{norm}_2(f^l(\mathcal{P}_i))) < \theta_2 \quad (7)$$

where n_i^l denotes the number of lower-level variables in $\mathcal{G}_i$; $\text{norm}_1(\mathcal{P}_i)$ and $\text{norm}_2(f^l(\mathcal{P}_i))$ represent the normalization of the variable values and lower-level objective function values of $\mathcal{P}_i$, respectively; $\text{ave}(\text{std}(\text{norm}_1(\mathcal{P}_i))))$ represents the average standard deviation of all dimensions of normalized $\mathcal{P}_i$; $\text{std}(\text{norm}_2(f^l(\mathcal{P}_i)))$ represents the standard deviation of the normalized lower-level objective function values of $\mathcal{P}_i$; and θ_1 and θ_2 are two parameters.

In terms of norm_1, the dth variable value of the jth individual in $\mathcal{P}_i$ at the lower level is normalized as

$$x_{j,d}^{l'} = \frac{x_{j,d}^l - x_d^{l,min}}{x_d^{l,max} - x_d^{l,min}} \quad (8)$$

where $x_d^{l,max}$ and $x_d^{l,min}$ represent the upper and lower bounds of the dth variable at the lower level, respectively.

In terms of norm_2, since the maximum lower-level objective function value in the search space is not easy to obtain, the log transformation [48] is adopted. The lower-level objective function value of the jth individual in $\mathcal{P}_i$ is normalized as

$$f_j^{l'} = log_{10}\left(f_j^l - \min(f^l(\mathcal{P}_i)) + \phi\right) \quad (9)$$

where $\min(f^l(\mathcal{P}_i))$ represents the recorded minimum lower-level objective function value of $\mathcal{P}_i$ during the evolution and ϕ is a small positive value (e.g., $1e-6$).

The main idea behind the judgment criterion in Eq. (7) is the following: if the difference of all individuals is larger than θ_1 in the decision space and smaller than θ_2 in the objective space, we consider that the subgroup has multiple optima.

In order to address the second issue, we design the following aggregation functions as new objective function and degree of constraint violation:

$$f'(\mathbf{x}^l) = \text{norm}_2(f^l(\mathbf{x}^l)) + \text{norm}_2(f^u(\mathbf{x}^l)) \quad (10)$$

$$cv'(\mathbf{x}^l) = \text{norm}_2(cv^l(\mathbf{x}^l)) + \text{norm}_2(cv^u(\mathbf{x}^l)) \quad (11)$$

where the log transformation is employed.

In Eqs. (10) and (11), since the information of the upper-level optimization problem is taken into account, the optimistic solution can be distinguished from all optima according to the definition of the optimistic solution.

The optimization of type-II subgroups is presented in **Algorithm 6**. It is similar to the optimization of type-I subgroups except for two modifications:

- The judgment criterion in Eq. (7) is adopted to judge whether the subgroup has multiple optima in line 15 of **Algorithm 6**.
- The aggregation functions in Eqs. (10) and (11) are used to evaluate the objective function values and degree of constraint violation of $\mathcal{O}_j^t$ if multiple optima exist, as shown in lines 11 and 17 of **Algorithm 6**.

3) Optimization of Type-III Subgroups: Since type-III subgroups contain both upper-level and lower-level variables, the single-level optimization algorithm is no longer applicable and the bilevel optimization algorithm is required. In this paper, we directly use the existing nested structure-based BLEA. Although we mention that the nested structure-based BLEA may suffer from severe computational burdens in large-scale BOPs, this issue has been eased in this paper due to the fact that each subgroup contains few variables.

Remark 1: The relationships between both levels have been studied in some papers, such as [24] and [49]. In [24], the initial distribution of the lower-level CMA-ES is generated by the marginal distribution of the upper-level CMA-ES, thus reducing the number of lower-level FEs. In [49], initial lower-level solutions are chosen according to the distances among upper-level solutions. This paper is different from [24] and [49] by taking the interactions between upper-level and lower-level variables into account. Under this condition, we can implement variable grouping in GO while [24] and [49] cannot.

III. Experimental Studies

This section introduces the experimental studies for investigating the performance of GO. First, we briefly describe the test problems and performance metrics. Then, five BLEAs for comparison are introduced. Subsequently, the parameter settings are presented. Finally, we give the results and discussions.

A. Test Problems and Performance Metrics

To evaluate the performance of different algorithms, a well-known scalable test suite, i.e., SMD [22], was employed, which consists of 12 problems and can be scaled to any number of variables. In our experiments, for each problem, three instances were considered: (10+10), (25+25), and (50+50) variables. According to [22], SMD has a variety of characteristics, such as linearity, convexity, disconnectedness, and multi-modality. Moreover, some problems in SMD have strong conflicts between both levels. These characteristics can test the multi-facet capabilities of BLEAs.

In this paper, three performance metrics were selected to evaluate the performance of the compared algorithms: the average normalized accuracy, the average number of FEs, and the average acceleration rate.

- In terms of the accuracy, we respectively calculated the absolute differences between the best objective function values provided by a BLEA and the true optimal objective function values at both levels, which are expressed as

$$Acc^u = \left| f^{u'} - f^{u^*} \right| \tag{12}$$

and

$$Acc^l = \left| f^{l'} - f^{l^*} \right| \tag{13}$$

where $f^{u'}$ and $f^{l'}$ are the best objective function values provided by a BLEA at both levels; and f^{u^*} and f^{l^*} are the true optimal objective function values at both levels. The true optimal objective function values of SMD are reported in [22]. It is worth noting that it is inappropriate to investigate the performance of an algorithm at each level separately [24]. For example, a good lower-level solution may lead to a bad upper-level solution. In contrast, a good upper-level solution may be accompanied by a bad lower-level solution. As a result, the accuracies at both levels should be considered at the same time. However, due to the different magnitudes of the accuracies at both levels, we respectively normalized them as follows:

$$Acc^{u'} = \frac{Acc^u - Acc^{u,min}}{Acc^{u,max} - Acc^{u,min}} \tag{14}$$

and

$$Acc^{l'} = \frac{Acc^l - Acc^{l,min}}{Acc^{l,max} - Acc^{l,min}} \tag{15}$$

where $Acc^{u,max}$ and $Acc^{u,min}$ represent the maximum and minimum accuracies obtained by the compared algorithms at the upper level on an instance over 30 runs, respectively; and $Acc^{l,max}$ and $Acc^{l,min}$ represent the maximum and minimum accuracies obtained by the compared algorithms at the lower level on an instance over 30 runs, respectively. Finally, the first performance metric is the sum of the average normalized accuracies over 30 runs: average($Acc^{u'}$) + average($Acc^{l'}$).

- Due to the nested structure, it is difficult to terminate all algorithms under the same number of FEs. Therefore, the numbers of FEs at both levels were also recorded, denoted as FEs^u and FEs^l. Similarly, the second performance metric is the sum of the average numbers of FEs at both levels over 30 runs: average(FEs^u) + average(FEs^l).
- The acceleration rate of algorithm B against algorithm A is calculated

$$AR_{ij} = \frac{(AFEs^u_{Aij} + AFEs^l_{Aij}) - (AFEs^u_{Bij} + AFEs^l_{Bij})}{AFEs^u_{Aij} + AFEs^l_{Aij}} \tag{16}$$

where $AFEs^u_{Aij}$ and $AFEs^l_{Aij}$ denote the average FEs^u and FEs^l of algorithm A on the jth instance of the ith problem, respectively; and $AFEs^u_{B,i}$ and $AFEs^l_{B,i}$ denote the average FEs^u and FEs^l of algorithm B on the jth instance of the ith problem, respectively. The third performance metric is $\frac{1}{36}\sum_{i=1}^{12}\sum_{j=1}^{3} AR_{ij}$.

TABLE II
Setting of the Population Size

Algorithm	Upper Level	Lower Level
NBLEA	100	100
GO-NBLEA	$min\{10*n^u_j, 100\}$	$min\{10*n^l_j, 100\}$
BIDE	30	30
GO-BIDE	$min\{10*n^u_j, 30\}$	$min\{10*n^l_j, 30\}$
BLEAQ	50	50
GO-BLEAQ	$min\{10*n^u_j, 50\}$	$min\{10*n^l_j, 50\}$
BLMA	50	50
GO-BLMA	$min\{10*n^u_j, 50\}$	$min\{10*n^l_j, 50\}$
BL-CMA-ES	$4+3\lfloor ln(n^u+n^l)\rfloor$	$4+3\lfloor ln(n^l)\rfloor$
GO-BL-CMA-ES	$4+3\lfloor ln(n^u_j+n^l_j)\rfloor$	$4+3\lfloor ln(n^l_j)\rfloor$

B. Algorithms for Comparison

The following five BLEAs using different EA paradigms were under our consideration for performance comparison.

- NBLEA [22]: NBLEA utilizes two identical steady-state single-objective GA to solve the upper-level and lower-level optimization problems.
- BIDE [21]: BIDE adopts DE as the search engine at each level.
- BLEAQ [25]: BLEAQ uses a steady-state GA at the upper level and combines a steady-state GA with an approximate model at the lower level.
- BLMA [23]: BLMA combines DE with a local search strategy at both levels.
- BL-CMA-ES [24]: BL-CMA-ES employs two identical CMA-ES to solve the upper-level and lower-level optimization problems.

In this paper, we embedded these five BLEAs into GO and their names were modified by adding "GO-". For example, the variant of NBLEA in GO was called GO-NBLEA. Note that, the variant of a BLEA in GO employed the the multi-start SQP and the single-level EA used in the original BLEA to optimize type-I and type-II subgroups, while type-III subgroups were directly optimized by the original BLEA.

TABLE III
SETTING OF THE NUMBER OF FES

$n^u + n^l$	$MaxFEs^u$	$MaxFEs^l$	$TotalFEs^l$
$10 + 10$	$5.00e + 3$	$5.00e + 2$	$2.50e + 6$
$25 + 25$	$8.00e + 3$	$7.00e + 2$	$5.60e + 6$
$50 + 50$	$1.20e + 4$	$1.00e + 3$	$1.20e + 7$

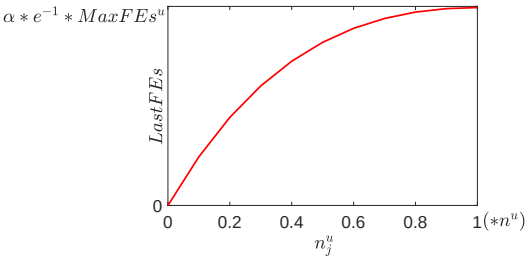


Fig. 3. Relationship between n_j^u and $LastFEs$.

C. Parameter Settings

Table II lists the detailed settings of the population size in different algorithms, where n_j^u and n_j^l represent the number of upper-level and lower-level variables in subgroup j, respectively. Due to the different numbers of variables, the subgroups have varied computational complexities; thus, different settings of the population size were employed for the subgroups in GO, which are related to the number of variables. In addition, the number of start points in the multi-start SQP was set to 7.

As introduced in Section II-B, if the maximum number of FEs is reached or the population has converged, the optimization of subgroups was terminated. The optimization of type-I subgroups stopped if the number of upper-level FEs exceeded $MaxFEs^u$. The optimization of type-II and type-III subgroups stopped if the number of upper-level FEs exceeded $MaxFEs^u$ or the total number of lower-level FEs exceeded $TotalFEs^l$. In addition, the lower-level optimization of type-III subgroups stopped for a given upper-level solution if the number of lower-level FEs exceeded $MaxFEs^l$. The settings of the number of FEs are summarized in Table III. The population was considered to have converged if the average standard deviation of all dimensions of the population in the decision space was less than $1e-4$ or the change range of the best objective function value during a certain number of FEs (denoted as $LastFEs$) was less than $1e-6$. Specifically, for multi-SQP, $LastFEs$ was set to the number of the start points. For EAs, in the optimization of type-I subgroups and the upper-level optimization of type-III subgroups, $LastFEs$ was set to

$$LastFEs = \alpha * (\frac{n_j^u}{n^u}) * exp(-\frac{n_j^u}{n^u}) * MaxFEs^u \quad (17)$$

and in the optimization of type-II subgroups and the lower level optimization of type-III subgroups, $LastFEs$ was set to

$$LastFEs = \alpha * (\frac{n_j^l}{n^l}) * exp(-\frac{n_j^l}{n^l}) * MaxFEs^l \quad (18)$$

where α is a constant scaling parameter. In this paper, α was set to 0.5.

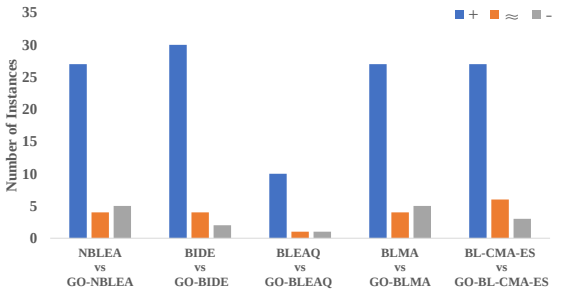
The reasons for designing the functions in Eqs. (17) and (18) are twofold. First, as the number of variables increases, finding a better solution requires more number of FEs. Therefore, in order to avoid the misjudging that the population has converged, $LastFEs$ increases with the increase of n_j^u or n_j^l. Second, to prevent from consuming too many FEs, the increase velocity of $LastFEs$ becomes slower as the number of variables increases. Fig. 3 shows the relationship between n_j^u and $LastFEs$.

In addition, ϵ was set to $1e-4$, and θ_1 and θ_2 were set to $1.6e-3$ and 1.2, respectively. Each algorithm was independently run 30 times for each instance of each problem. All the experiments were implemented in MATLAB and tested on a personal computer with Intel Core i7-7500 CPU @2.70 GHz and 8 GB of RAM. To test the statistical significance between two algorithms, the Wilcoxon's rank-sum test at a 0.05 significance level was conducted. The other parameter settings of BLEAs in GO were consistent with their original papers [21]–[25].

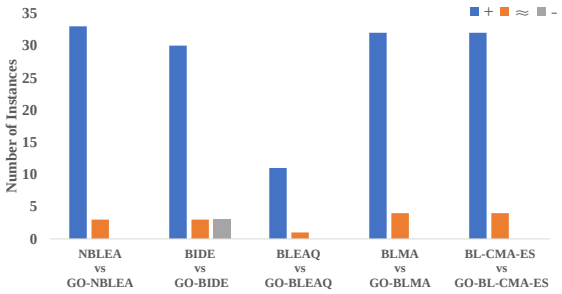
D. Comparison with NBLEA, BIDE, BLEAQ, BLMA, and BL-CMA-ES

Tables S-I–S-V in the supplementary file present the results of NBLEA and GO-NBLEA, BIDE and GO-BIDE, BLEAQ and GO-BLEAQ, BLMA and GO-BLMA, and BL-CMA-ES and GO-BL-CMA-ES regarding the average normalized accuracy and the average number of FEs over 30 runs on SMD, respectively. Note that, since the construction cost of the approximate model in BLEAQ increases significantly with the increase of the scale of BOPs, we only compared the performance of BLEAQ and GO-BLEAQ on the (10+10)-variable SMD. The better results in terms of the average normalized accuracy and the average number of FEs for each instance of each problem are highlighted with a gray background. The statistical test results between the original algorithms and their variants are summarized at the bottom of Tables S-I–S-V, where "+", "≈", and "−" represent that a variant performs significantly better than, equivalent to, and worse than its original algorithm, respectively. In addition, the average acceleration rate is also reported in Tables S-I–S-V.

From Tables S-I–S-V, GO-NBLEA, GO-BIDE, GO-BLEAQ, GO-BLMA, and GO-BL-CMA-ES show better performance than their original algorithms in terms of both the average normalized accuracy and the average number of FEs on SMD1–SMD8 and SMD11. It is worth noting that, on most of these instances, the variants can reduce the number of FEs by one to two orders of magnitude compared with the original algorithms. Moreover, GO-NBLEA, GO-BIDE, GO-BLEAQ, GO-BLMA, and GO-BL-CMA-ES are statistically better than their original algorithms on SMD1–SMD8 and SMD11. For SMD9, neither each original algorithm nor its variant can provide advantages in both the average normalized accuracy and the average number of FEs except that BIDE is superior to GO-BIDE on (25+25)-variable instance and GO-BL-CMA-ES is better than BL-CMA-ES on (10+10)-variable instance. However, according to the statistical test, there is actually no significant difference between each original algorithm and its variant on SMD9. Similarly, with respect to SMD10 and SMD12, each original algorithm and its variant cannot



(a)



(b)

Fig. 4. Wilcoxon's rank-sum test between the original algorithms (i.e., NBLEA, BIDE, BLEAQ, BLMA, and BL-CMA-ES) and their variants in GO (i.e., GO-NBLEA, GO-BIDE, GO-BLEAQ, GO-BLMA, and GO-BL-CMA-ES) in terms of the average normalized accuracy and the average number of FEs. (a) Average normalized accuracy. (b) Average number of FEs.

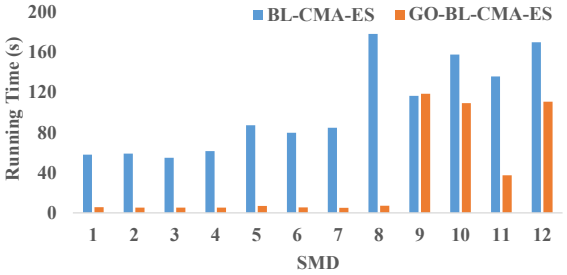


Fig. 5. Average running time of BL-CMA-ES and GO-BL-CMA-ES on (10+10)-variable SMD.

dominate each other on both the average normalized accuracy and the average number of FEs except that GO-NLBEA is better than NBLEA on (10+10)-variable SMD10, GO-BIDE has an edge over BIDE on (50+50)-variable SMD12, and GO-BLEAQ beats BLEAQ on (10+10)-variable SMD10. It can be observed that GO-NBLEA, GO-BIDE, GO-BLEAQ, GO-BLMA, and GO-BL-CMA-ES require fewer FEs on SMD12.

As shown in Fig. 4, according to the statistical test, in terms of the average normalized accuracy, GO-NBLEA, GO-BIDE, GO-BLEAQ, GO-BLMA, and GO-BL-CMA-ES outperform their original algorithms on 27, 30, 10, 27, and 27 instances, respectively, while lose on five, two, one, five, and three instances, respectively. In terms of the average number of FEs, GO-NBLEA, GO-BIDE, GO-BLEAQ, GO-BLMA, and GO-BL-CMA-ES perform better than their original algorithms on 33, 30, 11, 32, and 32 instances, respectively, while perform worse on zero, three, zero, zero, and zero instance, respectively. Therefore, we can conclude that, overall, GO is able to improve the convergence accuracy of BLEAs and reduce the number of FEs. Moreover, GO has the capability to reduce 72.89%, 56.95%, 69.95%, 66.50%, and 71.91% FEs for NBLEA, BIDE, BLEAQ, BLMA, and BL-CMA-ES, respectively.

Further, we would like to give the following comments:

- For SMD1–SMD8 and SMD11, GO breaks their variables into many small-scale subgroups. For example, the variables in SMD1 are divided into several one-dimensional type-I and type-II subgroups, and several two-dimensional type-III subgroups. The variables in SMD8 are divided into a multi-dimensional type-I subgroup, a multi-dimensional type-II subgroup, and several two-dimensional type-III subgroups. As for SMD11, its variables are divided into several one-dimensional type-I and type-II subgroups, and a multi-dimensional type-III subgroup. Compared with the original problems, these small-scale subgroups are easier to solve due to the small search space; thus, GO shows excellent performance on these problems. In addition, some problems (e.g., SMD2, SMD4–SMD8, and SMD11) have strong conflicts between both levels. The conflict is generally caused by a part of variables, and the changes in their values will lead to a positive effect at one level and a negative effect on the other level. In GO, the variables are divided into a set of subgroups. Under this condition, the conflict between both levels only affects the optimization of the type-III subgroups containing such variables; thus, GO can improve the performance of BLEAs on these problems.
- Regarding SMD9, all variables are interacting with each other. Under this condition, GO has to treat them as a whole group and, therefore, there is no significant difference between the original algorithm and its variant in terms of the average normalized accuracy. It is worth noting that GO still takes a certain number of FEs for variable grouping in the grouping phase. However, the number of FEs in the grouping phase is much smaller than that in the optimization phase. Thus, the performance difference between the original algorithm and its variant is also not significant in terms of the number of FEs.
- For SMD10 and SMD12, most variables are interacting due to the presence of constraints, resulting in only two subgroups. Although the dimension of the search space has been reduced, the search space of each subgroup in large-scale BOPs is still large. Therefore, GO can only improve the performance of BLEAs on a small number of instances.

In addition, we recorded the average running time of GO-BL-CMA-ES and BL-CMA-ES on (10+10)-variable SMD over 30 runs in Fig. 5. It is clear that GO-BL-CMA-ES is faster than its competitor except for SMD9, in which all variables are interacting with each other as mentioned before. Therefore, GO is capable of improving the efficiency of BLEAs on problems containing non-interacting variables.

TABLE IV
PERFORMANCE COMPARISON BETWEEN BL-CMA-ES AND GO-BL-CMA-ES REGARDING THE AVERAGE NORMALIZED ACCURACY AND THE AVERAGE NUMBER OF FES ON (100+100)-VARIABLE SMD.

$n^u + n^l$	Problem	BL-CMA-ES		GO-BL-CMA-ES	
		Average Normalized Accuracy	Average Number of FEs	Average Normalized Accuracy	Average Number of FEs
		Average($Acc^{u'}$) + Average($Acc^{l'}$)	Average(FEs^u) + Average(FEs^l)	Average($Acc^{u'}$) + Average($Acc^{l'}$)	Average(FEs^u) + Average(FEs^l)
$100+100$	SMD1	$8.04e-1+8.13e-1$ +	$1.25e+4+1.49e+7$ +	$4.80e-3+5.49e-3$	$1.91e+4+1.78e+5$
	SMD2	$7.10e-1+6.83e-1$ +	$1.37e+4+1.69e+7$ +	$5.80e-3+9.59e-3$	$1.87e+3+1.69e+5$
	SMD3	$7.01e-1+9.06e-1$ +	$1.34e+4+1.38e+7$ +	$3.66e-4+2.43e-3$	$1.99e+4+1.93e+5$
	SMD4	$5.44e-1+9.46e-1$ +	$1.34e+4+1.39e+7$ +	$5.82e-3+3.53e-3$	$1.83e+3+1.86e+5$
	SMD5	$8.32e-1+6.22e-1$ +	$3.00e+4+3.88e+7$ +	$2.42e-3+1.66e-3$	$2.38e+3+2.34e+5$
	SMD6	$6.71e-1+8.55e-1$ +	$1.57e+4+2.19e+7$ +	$7.30e-6+1.18e-4$	$1.92e+4+1.80e+5$
	SMD7	$9.62e-1+7.57e-1$ +	$3.00e+4+4.41e+7$ +	$1.51e-7+2.81e-8$	$2.59e+4+1.72e+5$
	SMD8	$9.86e-1+9.69e-1$ +	$3.00e+4+3.84e+7$ +	$1.70e-2+7.56e-3$	$2.64e+4+2.44e+5$
	SMD9	$5.28e-1+4.54e-1$ $\approx$	$3.00e+4+2.90e+7$ $\approx$	$2.71e-1+7.10e-1$	$3.00e+4+2.36e+7$
	SMD10	$6.30e-2+5.18e-1$ $\approx$	$3.00e+4+1.47e+7$ $\approx$	$2.11e-1+5.00e-1$	$3.00e+4+1.45e+7$
	SMD11	$8.73e-1+8.73e-1$ +	$3.00e+4+1.88e+7$ +	$6.49e-4+8.50e-10$	$3.00e+4+1.01e+7$
	SMD12	$2.91e-2+5.71e-2$ −	$3.00e+4+1.49e+7$ +	$7.15e-1+9.20e-1$	$3.00e+4+1.23e+7$
$+/\approx/-$		9/2/1	10/2/0		
Average Acceleration Rate					72.97%

E. Scalability of GO

To test the scalability of GO, we extended the problems to (100+100) variables and compared the performance of BL-CMA-ES and GO-BL-CMA-ES. The settings of the number of FEs are summarized as follows: $MaxFEs^u = 3.00e+4$, $MaxFEs^l = 1.50e+3$, and $TotalFEs^l = 4.50e+7$. Table IV shows the results of BL-CMA-ES and GO-BL-CMA-ES regarding the average normalized accuracy and the average number of FEs on (100+100)-variable SMD. It can be seen that GO-BL-CMA-ES provides better performance than BL-CMA-ES in terms of both the average normalized accuracy and the average number of FEs on SMD1–SMD9 and SMD11. As for SMD10 and SMD12, GO-BL-CMA-ES is worse than BL-CMA-ES in terms of the average normalized accuracy, whereas it is better than BL-CMA-ES in terms of the average number of FEs. According to the Wilcoxon's rank-sum test, GO-BL-CMA-ES outperforms BL-CMA-ES on nine and 10 instances in terms of the average normalized accuracy and the average number of FEs, respectively, while loses on one and zero instance, respectively. Besides, GO can reduce 72.97% FEs for BL-CMA-ES on (100+100)-variable SMD. The above experiments suggest that GO is still effective on larger-scale BOPs, which demonstrates the scalability of GO.

Remark 2: In the supplementary file, we also investigated the effect of the number of start points of the multi-start SQP, the effectiveness of the judgment criterion and the aggregation functions in the optimization of type-II subgroups, the grouping accuracy of the grouping phase, and the effectiveness of the multi-start SQP and the grouping phase.

IV. CASE STUDY: RESOURCE PRICING IN MEC

We consider a MEC system involving a MEC server managed by the MEC service provider (MSP) and a set of n mobile users denoted as $\mathcal{N} = \{1, \ldots, n\}$. Each mobile user has a delay-sensitive task to be executed. However, due to the computing capacity limitation of mobile devices, mobile users need to purchase computation resources from the MSP and then offload a part of their tasks to the MEC server for task execution.

1) System Model: We denote the task of mobile user i as M_i, which can be defined by a 3-tuple: $M_i = (D_i, X_i, T_i^{max}), \forall i \in \mathcal{N}$, where D_i and X_i denote the size of input data and the computation resources required to process a single bit in M_i, respectively, and T_i^{max} represents the maximum acceptable latency of M_i. Also, we define vector $\mathbf{o} = \{o_1, o_2, \ldots, o_n\}$ $(0 \le o_i \le 1, i \in \mathcal{N})$ as the offloading ratio. Based on o_i, M_i is divided into two parts: M_i^M and M_i^L, which are executed on the MEC server and the local mobile device, respectively. The size of the input data of M_i^M and M_i^L is $o_i D_i$ and $(1-o_i)D_i$, respectively.

In terms of M_i^M, it is first uploaded to the MEC server and then computed. The data rate of M_i^M is given as

$$R_i = B\log_2\left(1 + \frac{p_i d_i^{-\alpha}}{\sigma^2 + \sum_{m \in \mathcal{N}\setminus i} p_m d_m^{-\alpha}}\right), \ \forall i \in \mathcal{N} \quad (19)$$

where B denotes the system bandwidth, p_i is the transmission power of mobile device i [7], d_i represents the distance between mobile user i and the MEC server, α is the path loss exponent, and σ^2 is the white Gaussian noise power.

Then, the transmission time of M_i^M is given by

$$T_i^T = \frac{o_i D_i}{R_i}, \ \forall i \in \mathcal{N} \quad (20)$$

and the computing time of M_i^M is given by

$$T_i^C = \frac{o_i D_i X_i}{r_i^M}, \ \forall i \in \mathcal{N} \quad (21)$$

where r_i^M represents the amount of the computation resources purchased from the MSP. In Eq. (21), we define $T_i^C = 0$ for the case where $o_i = 0$ and $r_i^M = 0$.

Thus, the execution time of M_i^M can be written as [8]

$$T_i^M = T_i^T + T_i^C = \frac{o_i D_i}{R_i} + \frac{o_i D_i X_i}{r_i^M}, \ \forall i \in \mathcal{N} \quad (22)$$

In terms of M_i^L, the execution time is equal to the computing time, which is given by

$$T_i^L = \frac{(1-o_i) D_i X_i}{r_i^L}, \ \forall i \in \mathcal{N} \quad (23)$$

[7]For the sake of convenience, in this paper, the mobile device of mobile user i is called mobile device i.

[8]Because the size of the output data is generally much smaller than that of the input data, we omit the transmission time and energy consumption of the output data as in [50]–[52].

where r_i^L represents the computation capability of mobile device i.

To ensure that M_i is completed within $[0, T_i^{max}]$, the execution time of both M_i^M and M_i^L cannot be greater than T_i^{max}; therefore, one can get the following constraints:

$$0 \leq T_i^M \leq T_i^{max}, \ \forall i \in \mathcal{N} \tag{24}$$

and

$$0 \leq T_i^L \leq T_i^{max}, \ \forall i \in \mathcal{N} \tag{25}$$

If M_i can be completed under these two constraints, mobile user i can obtain the reward [9]. Meanwhile, mobile user i should pay for the computation resources purchased from the MSP and the energy consumption of mobile device i. The energy consumption of mobile device i consists of the energy consumption for transmitting M_i^M and computing M_i^L, which are expressed as

$$E_i^T = p_i T_i^T = \frac{p_i o_i D_i}{R_i} \tag{26}$$

and

$$E_i^L = k(r_i^L)^2(1-o_i)D_i X_i \tag{27}$$

where $k > 0$ is the effective capacitance coefficient.

Thus, the total profit of all mobile users can be expressed as

$$f^{MU}(\mathbf{o}, \mathbf{r}^M) = \sum_{i \in \mathcal{N}} \left(\theta - v_i^M r_i^M - v^L \left(E_i^T + E_i^L\right)\right) \tag{28}$$

where θ represents the reward that a mobile user receives after completing the task, $\mathbf{v}^M = \{v_1^M, \ldots, v_n^M\}$ represents the price set of the computation resources purchased from the MSP, and v^L denotes the price of the energy of the mobile device. Note that, in this paper, the discriminatory pricing scheme is considered where different prices of the computation resources are assigned to different mobile users [56].

Also, the MSP obtains the revenue by selling computation resources to mobile users but pays for the energy cost. We assume that the energy cost is related to the total computation resources for completing the tasks executed on the MEC server. Therefore, the profit of the MSP is expressed as

$$f^{MSP}(\mathbf{v}^M, \mathbf{o}, \mathbf{r}^M) = \sum_{i \in \mathcal{N}} \left(v_i^M r_i^M - v^E o_i D_i X_i\right) \tag{29}$$

where v^E denotes the price of energy consumed by the MSP.

2) Problem Formulation: In the MEC system, the MSP sets the price of computation resources, i.e., $\mathbf{v}^M$, with the aim of maximizing the profit, while the mobile users develop their strategies (the offloading ratio, i.e., $\mathbf{o}$, and the amount of computation resources, i.e., $\mathbf{r}^M$), based on the price announced by the MSP to maximize their total profit. Based on the behaviors of the MSP and the mobile users, we formulate the resource pricing problem in the MEC system as a BOP, which is formulated as

$$\begin{aligned}
\mathcal{P}: \max_{\mathbf{v}^M, \mathbf{o}, \mathbf{r}^M} & f^{MSP}(\mathbf{v}^M, \mathbf{o}, \mathbf{r}^M) \\
\text{s.t. } & v_{i,min}^M \leq v_i^M \leq v_{i,max}^M, \ i \in \mathcal{N} \\
& \{\mathbf{o}, \mathbf{r}^M\} \in \arg\max_{\mathbf{o}, \mathbf{r}^M} f^{MU}(\mathbf{o}, \mathbf{r}^M) \\
\text{s.t. } & 0 \leq T_i^M \leq T_i^{max}, \ i \in \mathcal{N} \\
& 0 \leq T_i^L \leq T_i^{max}, \ i \in \mathcal{N} \\
& 0 \leq o_i \leq 1, \ i \in \mathcal{N}, \\
& 0 \leq r^M \leq r_{i,max}^M, i \in \mathcal{N}
\end{aligned} \tag{30}$$

where $\mathbf{v}^M$ and $\{\mathbf{o}, \mathbf{r}^M\}$ are the upper-level and lower-level solutions, respectively; $v_{i,min}^M$ and $v_{i,max}^M$ denote the lowest and highest prices of the computation resources sold by the MSP to mobile user i, respectively; and $r_{i,min}^M$ and $r_{i,max}^M$ denote the maximum and minimum amount of computation resources sold by the MSP to mobile user i, respectively.

3) Parameter Settings: We assumed that 20 mobile users were randomly distributed in a 1000m×1000m square area and the MEC server was located in the center of the square area. Besides, D_i was randomly distributed within $[1, 10]$ KB, X_i was set to $1.8e+5$ cycles/bit, and T_i^{max} was set to 1s. α was assumed to be 4 and k was assumed to be 10^{-26}. B was set to 20 MHz, p_i was set to 0.1 W, and σ^2 was set to -174 dBm/Hz. Also, r_i^L was set to 1 Gcycles/s, and $r_{i,min}^M$ and $r_{i,max}^M$ were set to 0 and 10 Gcycles/s, respectively. θ was set to 100, v^L and v^E were set to 10 and 1 per Joule, respectively, and $v_{i,min}^M$ and $v_{i,max}^M$ were set to 1 and 20 per Joule, respectively. In addition, $MaxFEs^u$ was set to $8.00e+3$, $MaxFEs^l$ was set to $1.00e+3$, and $TotalFEs^l$ was set to $8.00e+6$.

4) Experimental Results: In this paper, we compared NBLEA, BIDE, and BL-CMA-ES with their variants in GO, respectively. Since the best known solution of the resource pricing problem in MEC is not available, the accuracy of the obtained solution cannot be computed. In this case, we measured the profit of the MSP rather than the accuracy. Specifically, each algorithm was first independently run 30 times. For a solution $\{\mathbf{v}^M, \mathbf{o}, \mathbf{r}^M\}$ obtained by an algorithm in one run, since $\{\mathbf{o}, \mathbf{r}^M\}$ may not be the optimal strategy corresponding to $\mathbf{v}^M$, it may lead to inaccurate evaluation of the profit of the MSP. To this end, we independently performed the lower-level optimization 30 times for $\mathbf{v}^M$ and recorded the best strategy $\{\mathbf{o}^*, \mathbf{r}^{M*}\}$ among all runs. Finally, based on $\{\mathbf{v}^M, \mathbf{o}^*, \mathbf{r}^{M*}\}$, the profit of the MSP was evaluated.

TABLE V
RESULTS OF SIX ALGORITHMS ON THE RESOURCE PRICING IN MEC

	Average Profit of the MSP	Average number of FEs
		Average(FEs^u) + Average(FEs^l)
NBLEA	$7.68e+1$	$7.39e+3+3.62e+6$
GO-NBLEA	$8.27e+1$	$8.00e+3+4.28e+5$
Average Acceleration Rate		87.98%
BIDE	$7.05e+1$	$3.78e+3+2.11e+6$
GO-BIDE	$7.76e+1$	$7.56e+3+6.76e+5$
Average Acceleration Rate		67.66%
BL-CMA-ES	$7.84e+1$	$8.00e+3+4.08e+6$
GO-BL-CMA-ES	$8.99e+1$	$8.00e+3+3.51e+5$
Average Acceleration Rate		91.22%

[9] M_i can be considered as the sensing task in mobile crowdsensing [53], the data collection task in Internet of things services [54], [55], or the mining task in mobile blockchain [56].

Table V gives the average profits of the MSP and the average numbers of FEs derived from the six algorithms over 30 runs. In addition, the average acceleration rate is also presented. From Table V, NBLEA, BIDE, and BL-CMA-ES exhibit poor average profits of the MSP and average numbers of FEs, compared with their variants in GO. To be specific, GO-NBLEA, GO-BIDE, and GO-BL-CMA-ES achieve 7.68%, 10.07%, and 14.67% improvements in the profit of the MSP, respectively. Also, they reduce 87.98%, 67.66%, and 91.22% FEs against their original algorithms, respectively. Therefore, the above experiments show the effectiveness of GO in solving a real-world BOP.

V. Conclusion

In this paper, we found that existing nested-based BLEAs may waste a large number of lower-level FEs to optimize the non-interacting lower-level variables when the value(s) of one/several upper-level variables change(s). Based on this finding, we proposed a new framework (called GO) to identify and utilize the interactions between upper-level and lower-level variables for scalable BOPs. GO consisted of two phases: the grouping phase and the optimization phase. The grouping phase divided upper-level and lower-level variables into three types of subgroups, i.e., types I-III, after identifying their interactions. In the optimization phase, the multi-start SQP and a single-level EA were employed to optimize type-I subgroups containing only upper-level variables and type-II subgroups containing only lower-level variables. Also, a BLEA was adopted to optimize type-III subgroups containing both upper-level and lower-level variables. Besides, after determining that a type-II subgroup has multiple optima by a criterion, we proposed new objective function and degree of constraint violation to guide the population toward the optimistic solution. The performance of GO was tested on a scalable test suite with up to (100+100) variables by embedding five representative BLEAs (i.e., NBLEA, BIDE, BLEAQ, BLMA, and BL-CMA-ES) into it. The results demonstrated the effectiveness of GO. Finally, GO was successfully applied to a practical case: the resource pricing in MEC.

Existing papers mainly research on the variable interactions in unconstrained optimization and rarely focus on the variable interactions caused by constraints. This paper proposes **Proposition** 2 to identify the variable interactions caused by constraints. In **Proposition 2**, if two variables are involved in the same constraint, they are considered to be interacting. However, there may be more effective ways to detect the variable interactions caused by constraints, which is a part of our future work. In addition, when dealing with bilevel combinatorial optimization problems, the main challenge in GO is that **Proposition** 1 and **Proposition** 2 are no longer applicable. In the future, we will study the interaction detection methods for bilevel combinatorial optimization and try to apply GO to large-scale bilevel combinatorial optimization problems [18], [27]. Extending GO to solve larger-scale BOPs (e.g., BOPs with more than (100+100) variables) is also one of our future studies.

References

[1] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: From classical to evolutionary approaches and applications," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 276–295, April 2018.

[2] J. Shu, R. Guan, L. Wu, and B. Han, "A bi-level approach for determining optimal dynamic retail electricity pricing of large industrial customers," *IEEE Transactions on Smart Grid*, vol. 10, no. 2, pp. 2267–2277, March 2019.

[3] V. Cardellini, V. Di Valerio, and F. Lo Presti, "Game-theoretic resource pricing and provisioning strategies in cloud systems," *IEEE Transactions on Services Computing*, to be published, doi: 10.1109/TSC.2016.2633266.

[4] J. Xie, Y. Mei, A. T. Ernst, X. Li, and A. Song, "A bi-level optimization model for grouping constrained storage location assignment problems," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 385–398, Jan 2018.

[5] Y. Wang, Z. Ru, K. Wang, and P. Huang, "Joint deployment and task scheduling optimization for large-scale mobile users in multi-uav-enabled mobile edge computing," *IEEE Transactions on Cybernetics*, to be published, doi: 10.1109/TCYB.2019.2935466.

[6] V. Suryan, A. Sinha, P. Malo, and K. Deb, "Handling inverse optimal control problems using evolutionary bilevel optimization," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, July 2016, pp. 1893–1900.

[7] S. Jenni and P. Favaro, "Deep bilevel learning," in *Proceedings of the 15th European Conference on Computer Vision (ECCV)*, September 2018, pp. 632–648.

[8] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, "Bilevel programming for hyperparameter optimization and meta-learning," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, July 2018, pp. 1563–1572.

[9] S. R. Hejazi, A. Memariani, G. Jahanshahloo, and M. M. Sepehri, "Linear bilevel programming solution by genetic algorithm," *Computers & Operations Research*, vol. 29, no. 13, pp. 1913–1925, 2002.

[10] Y. Jiang, X. Li, C. Huang, and X. Wu, "Application of particle swarm optimization based on CHKS smoothing function for solving nonlinear bilevel programming problem," *Applied Mathematics and Computation*, vol. 219, no. 9, pp. 4332–4339, 2013.

[11] Z. Wan, G. Wang, and B. Sun, "A hybrid intelligent algorithm by combining particle swarm optimization with chaos searching technique for solving nonlinear bilevel programming problems," *Swarm and Evolutionary Computation*, vol. 8, pp. 26–32, 2013.

[12] Z. Wan, L. Mao, and G. Wang, "Estimation of distribution algorithm for a class of nonlinear bilevel programming problems," *Information Sciences*, vol. 256, pp. 184–196, 2014.

[13] Y. Wang, Y.-C. Jiao, and H. Li, "An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 35, no. 2, pp. 221–232, 2005.

[14] A. Sinha, T. Soun, and K. Deb, "Using karush-kuhn-tucker proximity measure for solving bilevel optimization problems," *Swarm and Evolutionary Computation*, vol. 44, pp. 496–510, 2019.

[15] R. Mathieu, L. Pittard, and G. Anandalingam, "Genetic algorithm based approach to bi-level linear programming," *RAIRO-Operations Research*, vol. 28, no. 1, pp. 1–21, 1994.

[16] X. Zhu, Q. Yu, and X. Wang, "A hybrid differential evolution algorithm for solving nonlinear bilevel programming with linear constraints," in *2006 5th IEEE International Conference on Cognitive Informatics*, vol. 1. IEEE, 2006, pp. 126–131.

[17] P. Huang, Y. Wang, K. Wang, and Z. Liu, "A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing," *IEEE Transactions on Cybernetics*, to be published, doi: 10.1109/TCYB.2019.2916728.

[18] A. Chaabani, S. Bechikh, and L. Ben Said, "A co-evolutionary hybrid decomposition-based algorithm for bi-level combinatorial optimization problems," *Soft Computing*, to be published, doi:10.1007/s00500-019-04337-0.

[19] G. Zhang, G. Zhang, Y. Gao, and J. Lu, "Competitive strategic bidding optimization in electricity markets using bilevel programming and swarm technique," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 6, pp. 2138–2146, 2010.

[20] X. Li, P. Tian, and X. Min, "A hierarchical particle swarm optimization for solving bilevel programming problems," in *International Conference on Artificial Intelligence and Soft Computing*. Springer, 2006, pp. 1169–1178.

[21] J. S. Angelo, E. Krempser, and H. J. Barbosa, "Differential evolution for bilevel programming," in *2013 IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 470–477.

[22] A. Sinha, P. Malo, and K. Deb, "Test problem construction for single-objective bilevel optimization," *Evolutionary computation*, vol. 22, no. 3, pp. 439–477, 2014.

[23] M. M. Islam, H. K. Singh, T. Ray, and A. Sinha, "An enhanced memetic algorithm for single-objective bilevel optimization problems," *Evolutionary Computation*, vol. 25, no. 4, pp. 607–642, 2017.

[24] X. He, Y. Zhou, and Z. Chen, "Evolutionary bilevel optimization based on covariance matrix adaptation," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 258–272, April 2019.

[25] A. Sinha, P. Malo, and K. Deb, "Evolutionary algorithm for bilevel optimization using approximations of the lower level optimal solution mapping," *European Journal of Operational Research*, vol. 257, no. 2, pp. 395–411, 2017.

[26] M. M. Islam, H. K. Singh, and T. Ray, "A surrogate assisted approach for single-objective bilevel optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 681–696, Oct 2017.

[27] E. Kieffer, G. Danoy, M. R. Brust, P. Bouvry, and A. Nagih, "Tackling large-scale and combinatorial bi-level problems with a genetic programming hyper-heuristic," *IEEE Transactions on Evolutionary Computation*, to be published, doi: 10.1109/TEVC.2019.2906581.

[28] S. Dempe, *Bilevel optimization: theory, algorithms and applications*. TU Bergakademie Freiberg, Fakultät für Mathematik und Informatik, 2018.

[29] H. Önal, "A modified simplex approach for solving bilevel linear programming problems," *European Journal of Operational Research*, vol. 67, no. 1, pp. 126 – 135, 1993.

[30] J. Bard and J. Moore, "A branch and bound algorithm for the bilevel programming problem," *SIAM Journal on Scientific and Statistical Computing*, vol. 11, no. 2, pp. 281–292, 1990.

[31] L. N. Vicente and P. H. Calamai, "Bilevel and multilevel programming: A bibliography review," *Journal of Global Optimization*, vol. 5, no. 3, pp. 291–306, Oct 1994.

[32] E. Aiyoshi and K. Shimizu, "A solution method for the static constrained stackelberg problem via penalty method," *IEEE Transactions on Automatic Control*, vol. 29, no. 12, pp. 1111–1114, December 1984.

[33] A. Sinha, P. Malo, and K. Deb, "Evolutionary bilevel optimization: An introduction and recent advances," in *Recent Advances in Evolutionary Multi-objective Optimization*. Springer, 2017, pp. 71–103.

[34] H. Li, "A genetic algorithm using a finite search space for solving nonlinear/linear fractional bilevel programming problems," *Annals of Operations Research*, vol. 235, no. 1, pp. 543–558, 2015.

[35] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec 1997.

[36] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation," in *Proceedings of IEEE International Conference on Evolutionary Computation*, May 1996, pp. 312–317.

[37] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, "Benchmark functions for the CEC'2008 special session and competition on large-scale global optimization," Tech. Rep., 2009.

[38] A. Sinha, P. Malo, and K. Deb, "Efficient evolutionary algorithm for single-objective bilevel optimization," *arXiv preprint arXiv:1303.3901*, 2013.

[39] ——, "An improved bilevel evolutionary algorithm based on quadratic approximations," in *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 1870–1877.

[40] A. Ratle, "Kriging as a surrogate fitness landscape in evolutionary optimization," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 15, no. 1, p. 3749, 2001.

[41] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.

[42] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 378–393, 2013.

[43] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao, "DG2: A faster and more accurate differential grouping for large-scale black-box optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 6, pp. 929–942, 2017.

[44] X. Ma, F. Liu, Y. Qi, X. Wang, L. Li, L. Jiao, M. Yin, and M. Gong, "A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 2, pp. 275–298, 2015.

[45] Y. Wang, H. Liu, F. Wei, T. Zong, and X. Li, "Cooperative coevolution with formula-based variable grouping for large-scale global optimization," *Evolutionary Computation*, vol. 26, no. 4, pp. 569–596, 2018.

[46] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2, pp. 311 – 338, 2000.

[47] M. M. Islam, H. K. Singh, and T. Ray, "A surrogate assisted approach for single-objective bilevel optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 681–696, 2017.

[48] C. Feng, H. Wang, N. Lu, and X. M. Tu, "Log transformation: application and interpretation in biomedical research," *Statistics in Medicine*, vol. 32, no. 2, pp. 230–239, 2013.

[49] K. Deb and A. Sinha, "An efficient and accurate solution methodology for bilevel multi-objective programming problems using a hybrid evolutionary-local-search algorithm," *Evolutionary Computation*, vol. 18, no. 3, pp. 403–449, 2010.

[50] K. Wang, P. Huang, K. Yang, C. Pan, and J. Wang, "Unified offloading decision making and resource allocation in ME-RAN," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8159–8172, Aug 2019.

[51] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, October 2016.

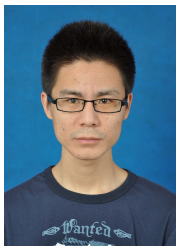
[52] K. Wang, K. Yang, and C. S. Magurawalage, "Joint energy minimization and resource allocation in C-RAN with mobile cloud," *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 760–770, July 2018.

[53] J. Hu, K. Yang, L. Hu, and K. Wang, "Reward-aided sensing task execution in mobile crowdsensing enabled by energy harvesting," *IEEE Access*, vol. 6, pp. 37 604–37 614, 2018.

[54] M. Moltafet, A. Rezaei, N. Mokari, M. R. Javan, H. Saeedi, and H. P. Nik, "Joint dynamic pricing and radio resource allocation framework for IoT services," *arXiv preprint arXiv:1903.02928*, 2019.

[55] P. Huang, Y. Wang, K. Wang, and K. Yang, "Differential evolution with a variable population size for deployment optimization in a UAV-assisted IoT data collection system," *IEEE Transactions on Emerging Topics in Computational Intelligence*, to be published, doi: 10.1109/TETCI.2019.2939373.

[56] Z. Xiong, S. Feng, W. Wang, D. Niyato, P. Wang, and Z. Han, "Cloud/fog computing resource management and pricing for blockchain networks," *IEEE Internet of Things Journal*, to be published, doi: 10.1109/JIOT.2018.2871706.



Pei-Qiu Huang received the B.S. degree in automation and the M.S. degree in control theory and control engineering both from the Northeastern University, Shenyang, China, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree in control science and engineering, Central South University, Changsha, China. His current research interests include evolutionary computation, bilevel optimization, and mobile edge computing.

Yong Wang (M'08–SM'17) received the Ph.D. degree in control science and engineering from the Central South University, Changsha, China, in 2011.

He is a Professor with the School of Automation, Central South University, Changsha, China. His current research interests include the theory, algorithm design, and interdisciplinary applications of computational intelligence.

Dr. Wang is an Associate Editor for the *IEEE Transactions on Evolutionary Computation* and the *Swarm and Evolutionary Computation*. He was a recipient of Cheung Kong Young Scholar in 2018 and a Web of Science highly cited researcher in Computer Science in 2017 and 2018.

Supplementary File for "A Framework for Scalable Bilevel Optimization: Identifying and Utilizing the Interactions between Upper-level and Lower-level Variables"

S-I. Supplementary Results

TABLE S-I
PERFORMANCE COMPARISON BETWEEN NBLEA AND GO-NBLEA REGARDING THE AVERAGE NORMALIZED ACCURACY AND THE AVERAGE NUMBER OF FES.

$n^u + n^l$	Problem	NBLEA		GO-NBLEA	
		Average Normalized Accuracy	Average Number of FEs	Average Normalized Accuracy	Average Number of FEs
		Average($Acc^{u'}$) + Average($Acc^{l'}$)	Average(FEs^u) + Average(FEs^l)	Average($Acc^{u'}$) + Average($Acc^{l'}$)	Average(FEs^u) + Average(FEs^l)
$10+10$	SMD1	$4.92e-1+3.28e-1$ +	$5.00e+3+1.04e+6$ +	$4.44e-4+6.48e-7$	$1.71e+3+6.60e+4$
	SMD2	$8.72e-1+6.80e-1$ +	$3.97e+3+1.03e+6$ +	$1.13e-3+1.14e-3$	$2.25e+3+9.68e+4$
	SMD3	$4.52e-1+7.35e-1$ +	$5.00e+3+1.08e+6$ +	$1.10e-4+2.00e-7$	$1.75e+3+6.88e+4$
	SMD4	$7.57e-1+7.95e-1$ +	$4.76e+3+1.05e+6$ +	$4.12e-4+4.98e-4$	$1.69e+3+7.00e+4$
	SMD5	$5.92e-1+6.83e-1$ +	$1.38e+3+4.23e+5$ +	$9.65e-6+2.30e-5$	$5.00e+3+7.96e+4$
	SMD6	$7.79e-1+6.33e-1$ +	$3.59e+3+1.04e+6$ +	$2.21e-3+3.40e-3$	$2.98e+3+1.16e+5$
	SMD7	$8.35e-1+4.49e-1$ +	$1.15e+3+3.04e+5$ +	$2.43e-3+2.60e-1$	$5.00e+3+1.04e+5$
	SMD8	$7.49e-1+7.48e-1$ +	$1.09e+3+3.35e+5$ +	$7.83e-3+9.11e-3$	$5.00e+3+9.74e+4$
	SMD9	$5.60e-1+2.66e-1$ ≈	$3.39e+3+7.47e+5$ ≈	$4.77e-1+4.99e-1$	$2.11e+3+4.61e+5$
	SMD10	$1.54e-1+5.84e-1$ ≈	$5.00e+3+1.08e+6$ +	$6.10e-1+5.07e-2$	$5.00e+3+4.27e+5$
	SMD11	$4.79e-1+7.50e-1$ +	$4.99e+3+1.06e+6$ +	$2.68e-1+6.54e-3$	$3.07e+3+4.78e+5$
	SMD12	$7.57e-2+2.14e-1$ −	$5.00e+3+1.03e+6$ +	$6.85e-1+5.22e-1$	$5.00e+3+3.97e+5$
$25+25$	SMD1	$5.85e-1+5.34e-1$ +	$8.00e+3+1.99e+6$ +	$2.00e-5+3.76e-7$	$4.41e+3+1.08e+5$
	SMD2	$6.04e-1+8.38e-1$ +	$7.86e+3+2.61e+6$ +	$6.38e-4+7.35e-4$	$4.47e+3+1.14e+5$
	SMD3	$6.68e-1+6.92e-1$ +	$8.00e+3+2.09e+6$ +	$2.51e-5+1.17e-7$	$4.61e+3+1.13e+5$
	SMD4	$7.08e-1+6.72e-1$ +	$8.00e+3+2.32e+6$ +	$1.13e-2+9.43e-3$	$4.36e+3+1.14e+5$
	SMD5	$7.67e-1+7.36e-1$ +	$3.14e+3+1.52e+6$ +	$1.03e-3+9.83e-4$	$8.00e+3+1.12e+5$
	SMD6	$6.01e-1+7.09e-1$ +	$7.60e+3+3.46e+6$ +	$3.08e-2+8.73e-3$	$6.43e+3+1.43e+5$
	SMD7	$6.01e-1+7.11e-1$ +	$2.85e+3+1.23e+6$ +	$2.87e-4+3.41e-4$	$7.15e+3+1.18e+5$
	SMD8	$6.97e-1+6.99e-1$ +	$1.90e+3+8.12e+5$ +	$1.94e-4+9.52e-4$	$8.00e+3+1.31e+5$
	SMD9	$5.00e-1+4.92e-1$ ≈	$7.20e+3+2.03e+6$ ≈	$6.58e-1+2.99e-2$	$7.58e+3+2.16e+6$
	SMD10	$1.22e-1+2.39e-1$ −	$8.00e+3+1.96e+6$ +	$6.00e-1+3.99e-1$	$7.93e+3+9.01e+5$
	SMD11	$4.32e-1+4.77e-1$ +	$8.00e+3+2.02e+6$ +	$1.23e-2+2.61e-3$	$7.73e+3+1.74e+6$
	SMD12	$1.80e-1+1.25e-1$ −	$8.00e+3+1.94e+6$ +	$6.76e-1+5.28e-1$	$8.00e+3+9.45e+5$
$50+50$	SMD1	$6.07e-1+5.40e-1$ +	$1.20e+4+3.69e+6$ +	$1.74e-5+2.17e-7$	$9.76e+3+1.99e+5$
	SMD2	$6.35e-1+8.24e-1$ +	$1.20e+4+5.09e+6$ +	$1.36e-3+3.57e-3$	$8.23e+3+1.60e+5$
	SMD3	$7.28e-1+7.54e-1$ +	$1.20e+4+3.87e+6$ +	$3.45e-5+8.33e-8$	$9.48e+3+2.03e+5$
	SMD4	$5.91e-1+6.07e-1$ +	$1.20e+4+4.24e+6$ +	$1.34e-2+9.61e-3$	$7.29e+3+1.44e+5$
	SMD5	$4.85e-1+5.00e-1$ +	$1.03e+4+7.15e+6$ +	$1.53e-2+1.41e-2$	$1.20e+4+1.67e+5$
	SMD6	$1.61e-1+7.64e-1$ +	$1.20e+4+7.14e+6$ +	$3.43e-1+1.14e-2$	$1.10e+4+1.87e+5$
	SMD7	$9.02e-1+7.31e-1$ +	$7.39e+3+4.75e+6$ +	$7.77e-4+2.97e-5$	$1.20e+4+1.66e+5$
	SMD8	$3.98e-1+3.99e-1$ +	$9.07e+3+6.22e+6$ +	$2.52e-3+2.84e-3$	$1.20e+4+1.57e+5$
	SMD9	$6.19e-1+4.73e-1$ ≈	$1.20e+4+4.78e+6$ ≈	$5.94e-1+5.66e-1$	$1.20e+4+4.60e+6$
	SMD10	$1.66e-1+3.83e-1$ −	$1.20e+4+3.57e+6$ +	$5.91e-1+4.59e-1$	$1.20e+4+1.64e+6$
	SMD11	$6.99e-1+7.19e-1$ +	$1.20e+4+3.70e+6$ +	$5.37e-3+8.26e-4$	$1.20e+4+3.43e+6$
	SMD12	$2.97e-1+4.09e-1$ −	$1.20e+4+3.55e+6$ +	$7.01e-1+6.53e-1$	$1.20e+4+1.80e+6$
+/≈/−		27/4/5	33/3/0		
Average Acceleration Rate					72.89%

TABLE S-II
PERFORMANCE COMPARISON BETWEEN BIDE AND GO-BIDE REGARDING THE AVERAGE NORMALIZED ACCURACY AND THE AVERAGE NUMBER OF FES.

$n^u + n^l$	Problem	BIDE		GO-BIDE	
		Average Normalized Accuracy	Average Number of FEs	Average Normalized Accuracy	Average Number of FEs
		Average($Acc^{u'}$) + Average($Acc^{l'}$)	Average(FEs^u) + Average(FEs^l)	Average($Acc^{u'}$) + Average($Acc^{l'}$)	Average(FEs^u) + Average(FEs^l)
$10+10$	SMD1	$5.62e-1+5.77e-1$ +	$3.29e+3+1.11e+6$ +	$4.47e-9+3.82e-9$	$2.08e+3+1.29e+5$
	SMD2	$7.99e-1+6.08e-1$ +	$1.79e+3+6.49e+5$ +	$5.69e-3+1.53e-3$	$2.57e+3+1.62e+5$
	SMD3	$7.63e-1+7.48e-1$ +	$3.29e+3+1.09e+6$ +	$1.09e-8+1.96e-9$	$2.18e+3+1.39e+5$
	SMD4	$7.84e-1+8.29e-1$ +	$2.00e+3+7.29e+5$ +	$3.18e-2+2.19e-2$	$2.58e+3+1.67e+5$
	SMD5	$6.93e-1+6.00e-1$ +	$1.76e+3+6.20e+5$ +	$4.65e-3+3.28e-3$	$5.00e+3+1.61e+5$
	SMD6	$6.13e-1+6.62e-1$ +	$2.56e+3+9.53e+5$ +	$3.97e-2+1.20e-2$	$3.51e+3+1.88e+5$
	SMD7	$7.01e-1+2.88e-1$ +	$1.94e+3+7.06e+5$ +	$4.52e-3+2.38e-3$	$4.77e+3+1.64e+5$
	SMD8	$6.20e-1+6.28e-1$ +	$1.55e+3+5.43e+5$ +	$1.58e-3+1.56e-3$	$5.00e+3+1.55e+5$
	SMD9	$4.86e-1+3.73e-1$ ≈	$2.11e+3+5.78e+5$ ≈	$2.76e-1+3.35e-1$	$2.19e+3+5.84e+5$
	SMD10	$2.50e-1+3.30e-1$ +	$2.16e+3+7.03e+5$ −	$3.58e-2+2.37e-2$	$3.31e+3+1.18e+6$
	SMD11	$3.15e-1+3.88e-1$ +	$2.30e+3+7.89e+5$ +	$4.78e-3+9.95e-5$	$2.15e+3+3.50e+5$
	SMD12	$1.05e-1+1.49e-1$ −	$3.12e+3+1.00e+6$ +	$7.09e-1+4.55e-1$	$3.00e+3+5.87e+5$
$25+25$	SMD1	$8.02e-1+7.23e-1$ +	$5.91e+3+2.73e+6$ +	$3.67e-8+2.35e-8$	$5.26e+3+1.97e+5$
	SMD2	$9.31e-1+8.72e-1$ +	$3.92e+3+2.21e+6$ +	$4.21e-3+8.10e-3$	$4.86e+3+1.87e+5$
	SMD3	$9.04e-1+8.82e-1$ +	$6.38e+3+2.86e+6$ +	$2.15e-8+1.66e-8$	$5.09e+3+2.04e+5$
	SMD4	$7.32e-1+7.59e-1$ +	$3.76e+3+2.04e+6$ +	$4.25e-3+2.55e-3$	$4.81e+3+1.91e+5$
	SMD5	$9.12e-1+9.07e-1$ +	$2.28e+3+1.15e+6$ +	$3.13e-3+2.82e-3$	$8.00e+3+1.94e+5$
	SMD6	$3.70e-1+8.15e-1$ +	$4.41e+3+2.41e+6$ +	$1.38e-1+2.26e-2$	$6.67e+3+1.86e+5$
	SMD7	$9.03e-1+5.15e-1$ +	$3.22e+3+1.74e+6$ +	$5.49e-3+8.82e-3$	$7.29e+3+1.59e+5$
	SMD8	$6.09e-1+6.11e-1$ +	$2.34e+3+1.17e+6$ +	$5.87e-3+5.80e-3$	$8.00e+3+1.78e+5$
	SMD9	$4.15e-1+3.53e-1$ ≈	$5.81e+3+1.60e+6$ ≈	$5.81e-1+2.85e-1$	$4.31e+3+1.67e+6$
	SMD10	$1.05e-1+1.05e-1$ +	$3.14e+3+1.50e+6$ −	$2.88e-7+2.74e-7$	$7.75e+3+4.21e+6$
	SMD11	$6.30e-1+6.91e-1$ +	$4.08e+3+2.22e+6$ +	$3.85e-3+4.09e-4$	$3.71e+3+9.45e+5$
	SMD12	$3.37e-1+3.16e-1$ −	$3.70e+3+1.77e+6$ +	$6.98e-1+5.78e-1$	$6.19e+3+1.49e+6$
$50+50$	SMD1	$8.04e-1+7.47e-1$ +	$1.04e+4+7.29e+6$ +	$3.58e-7+3.18e-7$	$1.11e+4+2.57e+5$
	SMD2	$6.89e-1+8.50e-1$ +	$6.23e+3+5.48e+6$ +	$6.39e-3+2.69e-2$	$9.26e+3+2.01e+5$
	SMD3	$6.68e-1+7.05e-1$ +	$9.94e+3+6.67e+6$ +	$4.01e-4+4.13e-4$	$1.12e+4+2.72e+5$
	SMD4	$8.59e-1+8.53e-1$ +	$6.93e+3+6.07e+6$ +	$5.28e-3+2.52e-3$	$9.77e+3+2.22e+5$
	SMD5	$7.99e-1+8.20e-1$ +	$4.20e+3+3.27e+6$ +	$8.24e-3+7.83e-3$	$1.04e+4+2.15e+5$
	SMD6	$3.26e-1+7.80e-1$ +	$6.01e+3+5.17e+6$ +	$1.32e-1+1.37e-2$	$1.19e+4+1.91e+5$
	SMD7	$8.55e-1+6.39e-1$ +	$4.64e+3+4.08e+6$ +	$3.61e-3+7.75e-4$	$1.20e+4+1.05e+5$
	SMD8	$8.11e-1+8.13e-1$ +	$4.26e+3+3.31e+6$ +	$6.42e-3+6.38e-3$	$1.20e+4+1.72e+5$
	SMD9	$5.96e-1+5.78e-1$ ≈	$7.02e+3+4.52e+6$ ≈	$7.39e-1+3.33e-1$	$7.53e+3+4.66e+6$
	SMD10	$1.73e-1+1.73e-1$ +	$5.06e+3+4.05e+6$ −	$9.07e-8+8.64e-8$	$1.20e+4+5.11e+6$
	SMD11	$7.16e-1+7.72e-1$ +	$8.39e+3+7.84e+6$ +	$5.28e-3+1.19e-4$	$8.02e+3+3.96e+6$
	SMD12	$5.23e-1+6.34e-1$ ≈	$6.53e+3+5.22e+6$ +	$5.50e-1+4.09e-1$	$7.16e+3+4.91e+6$
+/ ≈ /−		30/4/2	30/3/3		
Average Acceleration Rate					56.95%

TABLE S-III
PERFORMANCE COMPARISON BETWEEN BLEAQ AND GO-BLEAQ REGARDING THE AVERAGE NORMALIZED ACCURACY AND THE AVERAGE NUMBER OF FES.

$n^u + n^l$	Problem	BLEAQ		GO-BLEAQ	
		Average Normalized Accuracy	Average Number of FEs	Average Normalized Accuracy	Average Number of FEs
		Average($Acc^{u'}$) + Average($Acc^{l'}$)	Average(FEs^u) + Average(FEs^l)	Average($Acc^{u'}$) + Average($Acc^{l'}$)	Average(FEs^u) + Average(FEs^l)
$10+10$	SMD1	$3.35e-1+4.25e-1$ +	$5.00e+3+1.24e+6$ +	$3.03e-2+4.58e-5$	$1.28e+3+2.59e+4$
	SMD2	$8.32e-1+7.08e-1$ +	$4.63e+3+1.80e+6$ +	$6.99e-3+2.03e-3$	$1.94e+3+3.92e+4$
	SMD3	$3.00e-1+3.33e-1$ +	$4.97e+3+1.34e+6$ +	$1.73e-3+4.74e-7$	$1.40e+3+2.99e+4$
	SMD4	$2.33e-1+2.06e-1$ +	$4.63e+3+1.11e+6$ +	$1.27e-2+6.62e-3$	$1.72e+3+2.36e+4$
	SMD5	$3.76e-1+4.42e-1$ +	$2.33e+3+1.00e+6$ +	$3.08e-3+2.97e-3$	$5.00e+3+6.37e+4$
	SMD6	$3.89e-1+1.00e-1$ +	$4.59e+3+3.71e+5$ +	$4.57e-3+1.50e-3$	$1.79e+3+6.36e+3$
	SMD7	$7.44e-1+5.27e-1$ +	$1.71e+3+6.98e+5$ +	$2.74e-2+9.73e-2$	$4.90e+3+3.37e+5$
	SMD8	$3.90e-1+4.02e-1$ +	$2.43e+3+1.12e+6$ +	$4.58e-4+1.02e-3$	$5.00e+3+9.16e+5$
	SMD9	$7.83e-1+6.40e-1$ ≈	$4.33e+3+1.47e+6$ ≈	$5.16e-1+4.58e-1$	$5.00e+3+1.48e+6$
	SMD10	$4.02e-1+5.00e-1$ +	$5.00e+3+1.26e+6$ +	$8.06e-1+8.45e-2$	$5.00e+3+3.88e+5$
	SMD11	$1.61e-1+1.62e-1$ +	$5.00e+3+1.23e+6$ +	$1.61e-1+1.81e-3$	$3.95e+3+5.39e+5$
	SMD12	$1.01e-1+4.96e+1$ −	$4.76e+3+1.11e+6$ +	$5.35e-1+4.87e-1$	$5.00e+3+3.94e+5$
+/ ≈ /−		10/1/1	11/1/0		
Average Acceleration Rate					69.95%

TABLE S-IV
PERFORMANCE COMPARISON BETWEEN BLMA AND GO-BLMA REGARDING THE AVERAGE NORMALIZED ACCURACY AND THE AVERAGE NUMBER OF FES.

$n^u + n^l$	Problem	BLMA		GO-BLMA	
		Average Normalized Accuracy	Average Number of FEs	Average Normalized Accuracy	Average Number of FEs
		Average($Acc^{u'}$) + Average($Acc^{l'}$)	Average(FEs^u) + Average(FEs^l)	Average($Acc^{u'}$) + Average($Acc^{l'}$)	Average(FEs^u) + Average(FEs^l)
$10+10$	SMD1	$2.06e-1+2.10e-1$ +	$4.16e+3+1.37e+6$ +	$2.04e-3+1.05e-5$	$1.41e+3+8.56e+5$
	SMD2	$3.15e-1+4.97e-1$ +	$2.17e+3+8.47e+5$ +	$5.60e-6+5.82e-2$	$2.81e+3+1.98e+5$
	SMD3	$2.07e-1+2.03e-1$ +	$4.60e+3+1.51e+6$ +	$5.50e-5+2.37e-9$	$1.37e+3+8.48e+4$
	SMD4	$3.42e-1+4.57e-1$ +	$2.82e+3+1.05e+6$ +	$6.27e-11+1.96e-12$	$2.68e+3+1.86e+5$
	SMD5	$3.71e-1+3.25e-1$ +	$1.59e+3+6.01e+5$ +	$1.37e-4+3.88e-4$	$5.00e+3+1.56e+5$
	SMD6	$5.59e-1+8.84e-1$ +	$2.29e+3+9.20e+5$ +	$2.89e-2+1.31e-10$	$3.42e+3+1.96e+5$
	SMD7	$3.75e-1+4.54e-1$ +	$1.43e+3+5.59e+5$ +	$1.63e-3+5.38e-2$	$3.88e+3+1.80e+5$
	SMD8	$6.29e-1+4.51e-1$ +	$1.90e+3+7.11e+5$ +	$5.43e-3+3.62e-3$	$5.00e+3+1.10e+5$
	SMD9	$7.45e-1+1.86e-1$ ≈	$2.11e+3+6.36e+5$ ≈	$5.46e-1+3.78e-1$	$2.26e+3+6.76e+5$
	SMD10	$5.76e-1+1.22e-2$ −	$2.26e+3+7.82e+5$ +	$6.10e-1+9.12e-1$	$4.96e+3+5.87e+5$
	SMD11	$2.69e-1+3.70e-1$ +	$2.15e+3+7.63e+5$ +	$2.41e-2+1.05e-1$	$2.31e+3+3.48e+5$
	SMD12	$6.27e-2+2.94e-1$ −	$4.36e+3+1.11e+6$ +	$7.11e-1+9.47e-1$	$4.78e+3+5.44e+5$
$25+25$	SMD1	$2.00e-1+2.00e-1$ +	$6.74e+3+2.96e+6$ +	$1.18e-5+1.11e-10$	$4.11e+3+1.81e+5$
	SMD2	$7.31e-1+3.32e-1$ +	$4.01e+3+2.29e+6$ +	$4.54e-6+1.83e-3$	$5.18e+3+2.93e+5$
	SMD3	$3.12e-1+3.03e-1$ +	$6.75e+3+2.86e+6$ +	$2.93e-5+3.32e-8$	$4.33e+3+1.69e+5$
	SMD4	$5.72e-1+5.57e-1$ +	$3.68e+3+1.17e+6$ +	$1.49e-1+6.95e-2$	$6.73e+3+2.69e+5$
	SMD5	$5.21e-1+5.48e-1$ +	$3.36e+3+1.81e+6$ +	$2.93e-3+4.39e-3$	$8.00e+3+2.36e+5$
	SMD6	$7.50e-1+6.39e-1$ +	$3.14e+3+1.86e+6$ +	$1.34e-1+1.22e-8$	$7.03e+3+2.70e+5$
	SMD7	$6.01e-1+4.86e-1$ +	$2.32e+3+1.33e+6$ +	$3.07e-5+9.06e-5$	$7.40e+3+1.84e+5$
	SMD8	$5.05e-1+4.35e-1$ +	$2.42e+3+1.27e+6$ +	$2.36e-1+6.74e-2$	$8.00e+3+2.07e+5$
	SMD9	$4.22e-1+5.22e-1$ ≈	$2.87e+3+1.23e+6$ ≈	$4.96e-1+3.37e-1$	$4.02e+3+1.53e+6$
	SMD10	$8.69e-1+5.48e-2$ −	$2.91e+3+1.49e+6$ ≈	$4.18e-1+7.03e-1$	$8.00e+3+1.34e+6$
	SMD11	$2.16e-1+3.06e-1$ +	$6.77e+3+2.73e+6$ +	$1.38e-2+1.35e-3$	$4.98e+3+1.09e+6$
	SMD12	$2.14e-1+1.32e-1$ −	$6.23e+3+2.69e+6$ +	$7.01e-1+8.10e-1$	$8.00e+3+1.28e+6$
$50+50$	SMD1	$4.03e-1+3.99e-1$ +	$8.94e+3+5.53e+6$ +	$2.28e-3+5.21e-11$	$7.92e+3+2.36e+5$
	SMD2	$4.96e-1+7.33e-1$ +	$5.11e+3+4.59e+6$ +	$1.86e-6+1.43e-3$	$8.76e+3+4.70e+5$
	SMD3	$4.33e-1+4.15e-1$ +	$1.01e+4+5.86e+6$ +	$1.17e-6+6.32e-10$	$7.61e+3+1.99e+5$
	SMD4	$5.53e-1+8.41e-1$ +	$5.05e+3+4.72e+6$ +	$1.80e-2+1.47e-2$	$9.39e+3+3.03e+5$
	SMD5	$4.19e-1+5.71e-1$ +	$4.60e+3+3.68e+6$ +	$1.18e-1+1.09e-2$	$1.20e+4+2.90e+5$
	SMD6	$8.24e-1+7.47e-1$ +	$7.17e+3+5.98e+6$ +	$8.68e-2+3.68e-8$	$1.19e+4+3.34e+5$
	SMD7	$5.90e-1+5.63e-1$ +	$2.86e+3+2.54e+6$ +	$3.62e-5+2.09e-4$	$1.20e+4+1.93e+5$
	SMD8	$2.42e-1+2.43e-1$ +	$4.42e+3+3.63e+6$ +	$1.06e-1+7.49e-2$	$1.20e+4+2.52e+5$
	SMD9	$4.26e-1+5.20e-1$ ≈	$7.27e+3+4.64e+6$ ≈	$5.90e-1+3.68e-1$	$7.85e+3+4.02e+6$
	SMD10	$2.50e-1+2.61e-1$ ≈	$4.37e+3+3.63e+6$ +	$2.61e-1+3.22e-1$	$1.20e+4+2.50e+6$
	SMD11	$2.13e-1+2.41e-1$ +	$1.06e+4+7.22e+6$ +	$8.23e-3+3.64e-3$	$1.20e+4+3.89e+6$
	SMD12	$1.87e-2+6.09e-3$ −	$1.15e+4+6.91e+6$ +	$2.36e-1+2.21e-1$	$1.20e+4+3.26e+6$
$+/\approx/-$		27/4/5	32/4/0		
Average Acceleration Rate					66.50%

TABLE S-V
PERFORMANCE COMPARISON BETWEEN BL-CMA-ES AND GO-BL-CMA-ES REGARDING THE AVERAGE NORMALIZED ACCURACY AND THE AVERAGE NUMBER OF FES.

$n^u + n^l$	Problem	BL-CMA-ES		GO-BL-CMA-ES	
		Average Normalized Accuracy	Average Number of FEs	Average Normalized Accuracy	Average Number of FEs
		Average($Acc^{u'}$) + Average($Acc^{l'}$)	Average(FEs^u) + Average(FEs^l)	Average($Acc^{u'}$) + Average($Acc^{l'}$)	Average(FEs^u) + Average(FEs^l)
$10+10$	SMD1	$7.86e-1+7.63e-1$ +	$1.57e+3+5.42e+5$ +	$4.18e-3+2.22e-3$	$1.44e+3+4.31e+4$
	SMD2	$4.37e-1+3.51e-1$ +	$1.56e+3+5.44e+5$ +	$5.55e-2+1.29e-2$	$1.40e+3+4.12e+4$
	SMD3	$7.65e-1+6.76e-1$ +	$1.57e+3+4.80e+5$ +	$5.41e-3+4.43e-4$	$1.44e+3+4.24e+4$
	SMD4	$4.31e-1+5.87e-1$ +	$1.63e+3+5.38e+5$ +	$5.73e-3+4.19e-3$	$1.36e+3+3.95e+4$
	SMD5	$5.48e-1+4.21e-1$ +	$2.82e+3+8.27e+5$ +	$1.62e-1+2.37e-1$	$2.25e+3+6.53e+4$
	SMD6	$4.30e-1+3.71e-1$ +	$1.76e+3+6.38e+5$ +	$5.47e-2+1.25e-2$	$1.80e+3+4.51e+4$
	SMD7	$4.33e-1+5.78e-1$ +	$2.18e+3+6.98e+5$ +	$1.52e-1+8.04e-2$	$2.20e+3+3.98e+4$
	SMD8	$3.73e-1+2.02e-1$ +	$4.52e+3+1.58e+6$ +	$1.35e-2+3.76e-2$	$2.58e+3+7.11e+4$
	SMD9	$4.35e-1+3.07e-1$ ≈	$2.79e+3+7.98e+5$ ≈	$2.50e-1+9.73e-2$	$2.58e+3+7.17e+5$
	SMD10	$3.47e-1+3.37e-1$ ≈	$5.00e+3+1.11e+6$ +	$4.25e-1+3.74e-1$	$4.67e+3+7.14e+5$
	SMD11	$7.74e-1+7.68e-1$ +	$4.76e+3+9.48e+5$ +	$1.11e-2+1.55e-8$	$2.24e+3+3.08e+5$
	SMD12	$2.43e-1+3.93e-1$ −	$5.00e+3+1.15e+6$ +	$6.73e-1+5.69e-1$	$5.00e+3+8.01e+5$
$25+25$	SMD1	$8.87e-1+8.29e-1$ +	$3.48e+3+1.82e+6$ +	$3.22e-3+1.48e-3$	$3.48e+3+6.05e+4$
	SMD2	$3.53e-1+7.26e-1$ +	$3.51e+3+1.89e+6$ +	$3.51e-2+1.88e-2$	$3.62e+3+6.40e+4$
	SMD3	$8.35e-1+8.05e-1$ +	$3.61e+3+1.61e+6$ +	$3.20e-3+1.79e-4$	$3.73e+3+6.98e+4$
	SMD4	$2.00e-1+2.00e-1$ +	$4.18e+3+2.05e+6$ +	$8.29e-9+9.44e-9$	$3.53e+3+6.66e+4$
	SMD5	$2.44e-1+4.62e-1$ +	$7.11e+3+3.01e+6$ +	$3.84e-1+4.71e-2$	$4.82e+3+6.96e+4$
	SMD6	$5.29e-1+7.18e-1$ +	$4.07e+3+2.24e+6$ +	$8.94e-2+1.81e-2$	$4.82e+3+6.96e+4$
	SMD7	$1.74e-1+1.49e-1$ +	$8.00e+3+5.13e+6$ +	$1.93e-2+1.42e-2$	$6.62e+3+6.15e+4$
	SMD8	$7.57e-1+5.96e-1$ +	$8.00e+3+4.54e+6$ +	$7.61e-6+5.88e-6$	$5.78e+3+9.78e+4$
	SMD9	$9.20e-1+6.47e-1$ ≈	$6.65e+3+3.04e+6$ ≈	$8.16e-1+5.94e-1$	$6.74e+3+3.05e+6$
	SMD10	$4.03e-1+4.50e-1$ ≈	$8.00e+3+2.13e+6$ +	$6.26e-1+4.44e-1$	$7.47e+3+1.90e+6$
	SMD11	$7.00e-1+6.98e-1$ +	$8.00e+3+2.22e+6$ +	$3.42e-3+9.18e-10$	$6.87e+3+1.22e+6$
	SMD12	$6.07e-1+1.57e-1$ −	$8.00e+3+2.15e+6$ +	$5.50e-1+5.65e-1$	$8.00e+3+1.65e+6$
$50+50$	SMD1	$8.03e-1+8.18e-1$ +	$6.46e+3+5.16e+6$ +	$3.30e-3+6.41e-4$	$8.15e+3+1.18e+5$
	SMD2	$3.04e-1+7.86e-1$ +	$6.69e+3+5.55e+6$ +	$1.03e-1+3.43e-2$	$7.89e+3+1.11e+5$
	SMD3	$8.87e-1+8.55e-1$ +	$6.79e+3+4.84e+6$ +	$2.22e-3+1.67e-4$	$8.48e+3+1.23e+5$
	SMD4	$2.67e-1+2.66e-1$ +	$8.95e+3+6.96e+6$ +	$3.99e-9+3.98e-9$	$8.08e+3+1.15e+5$
	SMD5	$3.77e-1+6.33e-1$ +	$1.20e+4+9.80e+6$ +	$1.33e-2+2.91e-3$	$9.20e+3+1.42e+5$
	SMD6	$6.22e-1+7.60e-1$ +	$7.67e+3+6.74e+6$ +	$1.31e-1+1.33e-2$	$1.05e+4+1.15e+5$
	SMD7	$1.73e-1+1.39e-1$ +	$1.20e+4+1.19e+7$ +	$4.63e-2+2.41e-3$	$1.15e+4+1.05e+5$
	SMD8	$8.16e-1+6.95e-1$ +	$1.20e+4+1.03e+7$ +	$3.56e-6+1.59e-6$	$1.14e+4+1.54e+5$
	SMD9	$6.79e-1+5.64e-1$ ≈	$1.04e+4+6.00e+6$ ≈	$5.89e-1+4.45e-1$	$9.35e+3+6.32e+6$
	SMD10	$5.30e-1+3.54e-1$ ≈	$1.20e+4+4.63e+6$ ≈	$1.75e-1+6.79e-1$	$1.20e+4+4.71e+6$
	SMD11	$7.28e-1+7.27e-1$ +	$1.20e+4+5.35e+6$ +	$2.83e-3+4.97e-7$	$1.20e+4+2.84e+6$
	SMD12	$1.80e-1+1.19e+2$ −	$1.20e+4+4.58e+6$ +	$6.23e-1+7.01e-1$	$1.20e+4+3.41e+6$
+/ ≈ /−		27/6/3	32/4/0		
Average Acceleration Rate					71.91%

S-II. Additional Experiments and Discussions

A. Effect of the Number of Start Points of the Multi-Start SQP

TABLE S-VI
AVERAGE ACCURACY OF THE MULTI-START SQP WITH SEVEN DIFFERENT NUMBERS OF START POINTS ON (50+50)-VARIABLE SMD3 AND SMD4

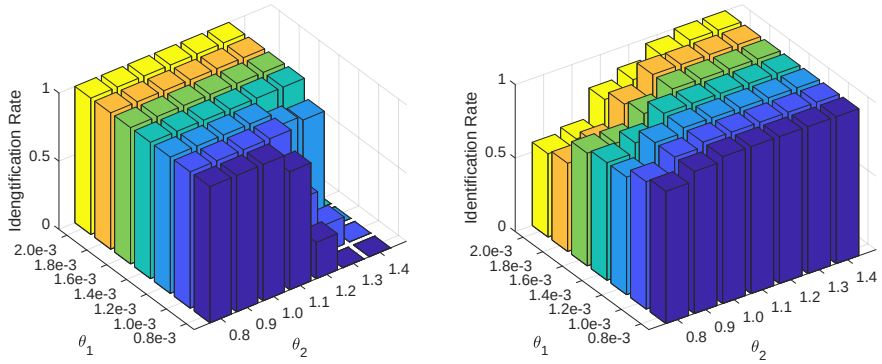
Population Size	(50+50)-variable SMD3	(50+50)-variable SMD4
	Average Accuracy	Average Accuracy
	Average(Acc^u) + Average(Acc^l)	Average(Acc^u) + Average(Acc^l)
One	$3.22e+1+3.42e+1$	$3.25e+1+3.44e+1$
Two	$5.06e+0+5.33e+0$	$5.95e+0+6.28e+0$
Three	$1.48e+0+1.55e+0$	$1.03e+0+1.08e+0$
Four	$6.98e-1+7.29e-1$	$6.03e-1+6.34e-1$
Five	$7.38e-2+6.59e-2$	$1.51e-1+1.59e-1$
Six	$6.03e-2+6.34e-2$	$3.01e-2+3.17e-2$
Seven	$5.90e-9+1.26e-9$	$1.47e-8+2.18e-8$

In this section, we investigated the effect of the number of start points of the multi-start SQP. We tested the multi-start SQP with seven different numbers of start points: one, two, three, four, five, six, and seven. (50+50)-variable SMD3 and SMD4 were employed for performance comparison since they have several local optima. In addition, GO-BL-CMA-ES was selected as the search algorithm. The results are given in Table S-VI. Note that, in order to better observe the average accuracy, we did not normalize it here. From Table S-VI, with the increase of the number of start points, the average accuracy is continuously improved. When the number of start points is greater than six, GO-BL-CMA-ES can find the global optimum. Therefore, seven is a reasonable choice for the number of start points of the multi-start SQP.

B. Effectiveness of the Judgment Criterion and the Aggregation Functions in the Optimization of Type-II Subgroups

TABLE S-VII
PERFORMANCE COMPARISON BETWEEN GO-BL-CMA-ES-WOJA AND GO-BL-CMA-ES REGARDING THE AVERAGE ACCURACY ON SMD6.

$n^u + n^l$	GO-BL-CMA-ES-WoJA	GO-BL-CMA-ES
	Average Accuracy	Average Accuracy
	Average(Acc^u) + Average(Acc^l)	Average(Acc^u) + Average(Acc^l)
$10 + 10$	$4.66e + 1 + 1.77e - 8$	$4.83e - 9 + 3.71e - 9$
$25 + 25$	$1.48e + 2 + 1.42e - 8$	$1.49e - 8 + 1.74e - 8$
$50 + 50$	$2.46e + 2 + 1.17e - 7$	$8.75e - 8 + 9.53e - 8$



(a) (10+10)-variable SMD5 (b) (10+10)-variable SMD6

Fig. S-1. Identification rate of the judgment criterion on (10+10)-variable SMD5 and SMD6 over 30 runs.

In the optimization of type-II subgroups, a criterion was proposed to judge whether a type-II subgroup has multiple optima or not. If multiple optima exist, two aggregation functions were adopted to guide the population toward the optimistic solution. Therefore, the judgment criterion and the aggregation functions play critical roles in the optimization of type-II subgroups. In order to verify their effectiveness, we developed a variant of GO-BL-CMA-ES, called GO-BL-CMA-ES-WoJA, where the judgment criterion and the aggregation functions were not used. Then, the average accuracy of GO-BL-CMA-ES was compared with that of GO-BL-CMA-ES-WoJA. SMD6 was selected for performance comparison due to the fact that its lower-level optimization problem is multi-modal and has multiple optima.

The results are summarized in Table S-VII. Note that, in order to better observe the average accuracy, we did not normalize it here. It is clear that the average accuracies of both GO-BL-CMA-ES and GO-BL-CMA-ES-WoJA are less than $1e - 6$ at the lower level of each instance, which means that both of them find the optimal solution at the lower level. However, for the upper-level optimization problem, the average accuracy of GO-BL-CMA-ES is much higher than that of GO-BL-CMA-ES-WoJA on each instance. This is because the optimal solution found by GO-BL-CMA-ES at the lower level is the optimistic solution, but the optimal solution found by GO-BL-CMA-ES-WoJA is not. The above experiments suggest that the judgment criterion and the aggregation functions are effective and can provide a guidance for the population to search for the optimistic solution.

We introduced two parameters (i.e., θ_1 and θ_2) in the judgment criterion to measure the difference of individuals in the decision space and the objective space, respectively. One may be interested in their effect on the performance of GO. To this end, we chose (10+10)-variable SMD5 and SMD6 as the test problems. The lower-level optimization problems of SMD5 and SMD6 are a unimodal problem and a multi-modal problem with infinite optima, respectively. We tested GO-BL-CMA-ES with 49 combinations of θ_1 and θ_2: $\theta_1 \in \{0.8e - 3, 1.0e - 3, \ldots, 2.0e - 3\}$ and $\theta_2 \in \{0.8, 0.9, \ldots, 1.4\}$, and recorded the identification rate over 30 runs in Fig. S-1. The identification rate indicates the proportion of runs in which GO-BL-CMA-ES can successfully identify that SMD5 does not have any type-II subgroup with multiple optima and SMD6 does have some type-II subgroups with multiple optima.

As shown in Fig. S-1, the judgment criterion performs well in the case of $\theta_1 \geq 1.4e - 3$ and $\theta_2 \leq 1.3$ on (10+10)-variable SMD5. On the contrary, the judgment criterion performs well on (10+10)-variable SMD6 when $\theta_1 \leq 1.8e - 3$ and $\theta_2 \geq 1.1$. Therefore, $\theta_1 = 1.6e - 3$ and $\theta_2 = 1.2$ are recommended in this paper.

C. Grouping Accuracy of the Grouping Phase

TABLE S-VIII
GROUPING ACCURACY OF THE GROUPING PHASE ON (50+50)-VARIABLE SMD OVER 30 RUNS.

Problem	SMD1	SMD2	SMD3	SMD4	SMD5	SMD6	SMD7	SMD8	SMD9	SMD10	SMD11	SMD12
Grouping Accuracy	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

In order to evaluate the grouping accuracy of the grouping phase, we first analytically derived the interactions between upper-level and lower-level variables, and then obtained the grouping results. Subsequently, they were compared with those obtained from the grouping phase, and the grouping accuracies of the grouping phase on (50+50)-variable SMD over 30 runs are recorded in Table S-VIII. The grouping accuracy indicates the proportion of runs that the grouping results obtained from the grouping phase are the same as those provided by the analytical derivation. From Table S-VIII, it is clear that the grouping phase is able to implement variable grouping accurately.

D. Effectiveness of the Multi-start SQP and the Grouping Phase

TABLE S-IX
PERFORMANCE COMPARISON AMONG GO-BL-CMA-ES, BL-CMA-ES, AND GO-BL-CMA-ES-WOS REGARDING THE AVERAGE NORMALIZED ACCURACY AND THE AVERAGE NUMBER OF FES.

$n^u + n^l$	Problem	GO-BL-CMA-ES			BL-CMA-ES			GO-BL-CMA-ES-WoS		
		Average Normalized Accuracy	Average Number of FEs		Average Normalized Accuracy	Average Number of FEs		Average Normalized Accuracy	Average Number of FEs	
		Average($Acc^{u'}$) + Average($Acc^{l'}$)	Average(FEs^u) + Average(FEs^l)		Average($Acc^{u'}$) + Average($Acc^{l'}$)	Average(FEs^u) + Average(FEs^l)		Average($Acc^{u'}$) + Average($Acc^{l'}$)	Average(FEs^u) + Average(FEs^l)	
$10+10$	SMD1	$4.23e-3+2.29e-3$	$1.44e+3+4.31e+4$	$-$	$7.86e-1+7.63e-1$	$1.57e+3+5.42e+5$	$+$	$4.32e-1+5.87e-1$	$1.91e+3+4.49e+4$	
	SMD2	$5.54e-2+1.29e-2$	$1.40e+3+4.12e+4$	$-$	$4.34e-1+3.52e-1$	$1.56e+3+5.44e+5$	$+$	$2.85e-1+1.49e-1$	$1.88e+3+4.32e+4$	
	SMD3	$4.51e-3+7.90e-5$	$1.44e+3+4.24e+4$	$\approx$	$6.45e-1+1.21e-1$	$1.57e+3+4.80e+5$	$+$	$3.98e-1+3.17e-1$	$1.91e+3+4.15e+4$	
	SMD4	$5.41e-3+3.12e-3$	$1.36e+3+3.95e+4$	$+$	$4.31e-1+5.42e-1$	$1.63e+3+5.38e+5$	$+$	$5.12e-2+4.72e-1$	$1.84e+3+3.84e+4$	
	SMD5	$2.63e-1+1.44e-1$	$2.25e+3+6.53e+4$	$-$	$6.03e-1+2.56e-1$	$2.82e+3+8.27e+5$	$+$	$2.30e-1+4.86e-1$	$2.43e+3+6.73e+4$	
	SMD6	$5.47e-2+1.25e-2$	$1.80e+3+4.51e+4$	$\approx$	$4.30e-1+3.71e-1$	$1.81e+3+6.38e+5$	$+$	$2.59e-1+1.67e-1$	$2.29e+3+4.60e+4$	
	SMD7	$1.52e-1+8.04e-2$	$2.20e+3+3.98e+4$	$\approx$	$4.33e-1+5.28e-1$	$2.18e+3+6.98e+5$	$+$	$1.18e-1+1.13e-1$	$1.95e+3+4.06e+4$	
	SMD11	$1.32e-2+1.56e-8$	$2.24e+3+3.08e+5$	$+$	$7.75e-1+7.68e-1$	$4.76e+3+9.48e+5$	$+$	$7.41e-3+6.02e-3$	$5.00e+3+1.53e+5$	
$+/\approx/-$		$8/0/0$	$3/3/2$		$6/2/0$	$8/0/0$				

In order to verify the effectiveness of the multi-start SQP and the grouping phase, the multi-start SQP in GO-BL-CMA-ES was replaced with CMA-ES, resulting in a variant called GO-BL-CMA-ES-WoS. SMD1–SMD6, SMD8, and SMD10 were selected for performance comparison since the multi-start SQP is used in GO-BL-CMA-ES when solving these problems. In this section, we considered (10+10) variables. Table S-IX presents the results of GO-BL-CMA-ES, BL-CMA-ES, and GO-BL-CMA-ES-WoS regarding the average normalized accuracy and the average number of FEs.

As shown in Table S-IX, GO-BL-CMA-ES is better than GO-BL-CMA-ES-WoS in terms of the average normalized accuracy on all problems. In addition, it requires fewer FEs on SMD1, SMD2, and SMD5–SMD7 compared with GO-BL-CMA-ES-WoS. However, GO-BL-CMA-ES-WoS is superior to GO-BL-CMA-ES only in terms of the average number of FEs on SMD3, SMD4, and SMD11. The above results suggest that the multi-start SQP can improve the performance of GO. On the other hand, GO-BL-CMA-ES-WoS is superior to BL-CMA-ES in terms of both the average normalized accuracy and the average number of FEs except for SMD3 and SMD5. Although the performance of GO-BL-CMA-ES-WoS is not significantly different from that of BL-CMA-ES in terms of the average normalized accuracy on SMD3 and SMD5, GO-BL-CMA-ES-WoS requires fewer FEs on these two problems. Note that the main difference between GO-BL-CMA-ES-WoS and BL-CMA-ES is that the former performs variable grouping but the latter does not. Therefore, the effectiveness of the grouping phase has been demonstrated.