Soft Comput
DOI 10.1007/s00500-016-2060-y

METHODOLOGIES AND APPLICATION

Multilevel framework for large-scale global optimization

Sedigheh Mahdavi[2] · **Shahryar Rahnamayan**[2] · **Mohammad Ebrahim Shiri**[1]

Abstract Large-scale global optimization (LSGO) algorithms are crucially important to handle real-world problems. Recently, cooperative co-evolution (CC) algorithms have successfully been applied for solving many large-scale practical problems. Many applications have imbalanced subcomponents where the size of subcomponents and their contribution to the objective function value are different. CC algorithms often lose their efficiency on LSGO problems with the imbalanced subcomponents; since they do not consider the imbalance aspect of variables. In this paper, we propose a multilevel optimization framework based on variables effect (called MOFBVE) which optimizes several subcomponents of the most important variables at earlier stages of optimization procedure before optimizing the problem with the original search space at its last stage. Sensitivity analysis (SA) method determines how the variation in the outputs of the model can be influenced by the variation of its input parameters. MOFBVE computes the main effect of variables using an SA method, Morris screening, and then it employs the k-means clustering method to construct groups including variables with the similar effects on the fitness value. The constructed groups are sorted in the descending order based on their contribution on the fitness value and the top groups are selected as the levels of the important variables. MOFBVE can reduce the complexity of search space to work with a simplified model to achieve an efficient exploration. The performance of MOFBVE is benchmarked on the imbalanced LSGO problems, i.e., two individually modified CEC-2010 and the CEC-2013 LSGO benchmark functions. The simulated experiments confirmed that MOFBVE obtains a promising performance on the majority of the imbalanced LSGO test functions. Also, MOFBVE is compared with state-of-the-art CC algorithms; and the results show that it is better than or at least comparable to CC algorithms.

Keywords Large-scale global optimization (LSGO) · Cooperative co-evolution (CC) · Sensitivity analysis (SA) · Multilevel · Variable effect

Communicated by V. Loia.

✉ Shahryar Rahnamayan
Shahryar.Rahnamayan@uoit.ca

Sedigheh Mahdavi
sedigheh.mahdavi@uoit.ca

[1] Department of Mathematics and Computer Science, Amirkabir University of Technology, Tehran, Iran

[2] Department of Electrical, Computer, and Software Engineering, University of Ontario Institute of Technology (UOIT), 2000 Simcoe Street North, Oshawa, ON L1H 7K4, Canada

1 Introduction

Large-scale global optimization (LSGO) is a demanding research direction because the majority of scientific and engineering problems with a large number of decision variables (such as designing large-scale electronic systems, scheduling problems with the large number of resources and service providers, vehicle routing in the large-scale traffic networks, gene recognition in the bioinformatics, inverse problem chemical kinetics, etc.) are formulated as LSGO problems. Recently, several metaheuristic algorithms have been applied to solve LSGO problems, but their performance deteriorates rapidly as the dimension of problems increases (Li et al. 2013; Tang et al. 2010; Tenne and Goh 2010). Recently, a large number of enhanced metaheuristic algorithms have been proposed to solve LSGO problems efficiently. In general, for tackling LSGO problems, there are two main branches

Published online: 12 February 2016

of metaheuristic algorithms: cooperative co-evolution (CC) algorithms (Potter and De Jong 1994; Potter 1997), and non-decomposition-based methods. CC algorithms decompose LSGO problems into several low-dimensional subcomponents in order to solve them using divide-and-conquer approach. Due to interactions among variables in the nonseparable problems, the majority of the decomposition methods attempts to recognize interacting variables and assign them to the same subcomponent since it has a significant impact on the efficiency of the optimization processes. The non-decomposition-based methods enhanced the performance of standard metaheuristic algorithms to tackle LSGO problems by focusing on especial alterations such as defining new mutation, selection, and crossover operators, designing and using local search, opposition-based learning, sampling operator, hybridization, or variable population size methods (Mahdavi et al. 2015). For more information on LSGO algorithms, the reader is referred to Mahdavi et al. (2015); a survey paper which has been published recently, covering the CC and non-decomposition-based methods to solve LSGO problems.

In the majority of real-world problems, their subcomponents are imbalanced, i.e., the size of subcomponents and their contribution to the objective function value are different (Li et al. 2013; Omidvar et al. 2015). While many research works have been conducted to tackle LSGO problems, only a few works have been reported to solve imbalanced LSGO problems. In Omidvar et al. (2011), the modified CC algorithms were proposed which assign more computational budget to the subcomponent with the maximum effect on the objective function. Many existing LSGO algorithms do not pay enough attention to the imbalance aspect of variables and thus lose their efficiency on imbalanced LSGO problems. Therefore, studying and considering the different effects of all variables in the imbalanced LSGO problems would be beneficial to propose efficient LSGO algorithms. Our study is motivated by two fundamental questions: (1) how can we identify the effect of variables in a black-box function (i.e., no a priori knowledge about the problem is given)? (2) How LSGO algorithms can use the information about variables' effect to improve their performance? This paper attempts to address these questions.

Sensitivity analysis methods allow to understand the relationships and impacts of input parameters on the output of a model (Saltelli et al. 2000, 2008; Rabitz and Aliş 1999). We used the sensitivity analysis methods to rank the importance of the variables in terms of their influence on the objective function value. In this paper, we propose a multilevel optimization framework based on variables effect (MOFBVE). In MOFBVE, the levels of significant variables are constructed to obtain a simplified model of LSGO problem in which variables with the most influence on the objective function are optimized and the values of unimportant variables, i.e., variables with less effect are fixed. By constructing the smaller levels of the search space, the proposed method reduces the complexity of the search space in LSGO problems at first stages to obtain suitable candidate solutions as initial solutions at next stages. The performance of MOFBVE is evaluated on the set of benchmark functions which are the modified version of CEC-2010 benchmark problems with the imbalanced subcomponents and the CEC-2013 LSGO benchmark functions. Compared to other CC algorithms, the simulated experiments demonstrate that MOFBVE is a highly competitive optimization algorithm for solving LSGO problems.

The organization of the rest of this paper is as follows. Section 2 gives a background review of CC algorithms and Morris screening method. Section 3 describes the proposed CC framework in detail. Section 4 presents the experimental results and discussion. Finally, the paper is concluded in Sect. 5.

2 Background review

2.1 Cooperative co-evolution

In 1994, the CC algorithms were proposed by Potter and De Jong (1994); Potter (1997). The classical steps of CC algorithms can be summarized as follows:

- Decompose a high-dimensional objective vector into some low-dimensional subcomponents.
- Evolve each subcomponent by a traditional optimization algorithm for a predefined number of generations using a round-robin strategy.
- Merge the solutions of all subcomponents to construct n-dimensional solutions to evaluate the individuals in each of the subcomponents.
- Stop the evolutionary process if termination condition is satisfied.

Over the past decades, various metaheuristic optimization algorithms including evolutionary programming (Liu et al. 2001), particle swarm optimization (PSO) (Van den Bergh and Engelbrecht 2004; Sun et al. 2012; Li and Yao 2012, 2009), artificial bee colony (ABC) (Ren and Wu 2013; Chen et al. 2008), evolutionary algorithms (EAs) (Liu et al. 2001; Singh and Ray 2010), and differential evolution (DE) (Yang et al. 2008a, b; Chen et al. 2010) have been incorporated into the CC framework for tackling LSGO problems. The major challenge of CC algorithms is developing an appropriate decomposition method to construct the subcomponents with the minimum interdependency according to the interaction among variables (Mahdavi et al. 2015; Chen et al. 2010; Omidvar et al. 2014a). Also, several CC algorithms with different decomposition strategies were proposed which

can assign the interacting variables into the same subcomponents and show an excellent performance such as MLCC (Yang et al. 2008b), DECC-G (Yang et al. 2008a), CCEA-AVP (Singh and Ray 2010), CCVIL (Chen et al. 2010), CCOABC (Ren and Wu 2013), CC-CMA-ES (Liu and Tang 2013), DECC-DG (Omidvar et al. 2014a), HDIMA (Sayed et al. 2012b), DIMA (Sayed et al. 2012a), DM-HDMR (Mahdavi et al. 2014), and MLSoft (Omidvar et al. 2014b), etc. For more detailed information about these methods and their challenge to decompose LSGO problems, please refer to (Mahdavi et al. 2015).

Most of the CC and decomposition methods have been evaluated only on LSGO problems with the equal balanced subcomponents while the most real-world problems contain imbalance subcomponents (Li et al. 2013; Omidvar et al. 2015, 2014a). Recently, several design features were proposed in Omidvar et al. (2015) to develop large-scale optimization benchmark suites for better approximating real-world problems of which the imbalance among subcomponents of an LSGO problem is one of these features. A major challenge part of CC algorithms in solving imbalanced LSGO problems lies in the decomposition step. Recently, some decomposition strategies by the automatic identification of variable interactions and high accuracy were proposed on LSGO problems with balanced subcomponents. Chen et al. (2010) proposed a decomposition method (CCVIL) based on the changes of the fitness function value among variables which can adaptively construct subcomponents by learning the interaction among the decision variables. Since the imbalanced LSGO problems have some variables with the significant influence and some variables with the small influence on the fitness value, CCVIL cannot correctly compare the fitness values to detect interaction among variables and also, it needs an excessive search in the learning step. Omidvar et al. (2014a) introduced the differential grouping approach (DECC-DG) which compares the approximation of the gradient to detect interactions among variables. The performance of DECC-DG decreases to decompose imbalanced LSGO problems (Omidvar et al. 2015) since it fails to compare the changes of the fitness values on the problems including variables with the various influences. In Mahdavi et al. (2014), a decomposition method, DM-HDMR, was proposed based on the obtained correlations of high-dimensional model representation (first-order RBF-HDMR). The first-order RBF-HDMR cannot correctly model the imbalanced functions therefore DM-HDMR has the low accuracy to detect the interaction among variables on the imbalanced LSGO problems. Therefore, finding an optimal decomposition method for the LSGO problems becomes extremely complicated when there are some imbalance subcomponents among them.

On the other hand, CC algorithms divide the computational budget equally among all subcomponents, therefore they waste the computational budget for optimizing subcomponents with the low effect on the global fitness (Omidvar et al. 2011, 2014a). In such scenarios, if decomposition method can obtain a near-ideal decomposition of the decision variables, CC algorithms still lose their efficiency to tackle with imbalanced LSGO problems. In Omidvar et al. (2011, 2014a), a contribution-based cooperative coevolution (CBCC) method was proposed which spends more computational budget on the subcomponents with the maximum influence on the global fitness.

2.2 Sensitivity analysis

The computational models of real-world problems are often complex, computationally expensive, with many input parameters, and non-linear relations. Sensitivity analysis methods are very powerful tools to study relationships among model parameters and identify the significant input parameters with the most impact on the outputs of the model. Saltelli et al. (2008) categorized sensitivity analysis methods into three main classes: screening, local, and global methods. Local methods provide information on the model behavior only around a reference point. Global sensitivity analysis provides information on how the entire range of the variation in the model outputs can be influenced by the variation of its input parameters. Screen methods can detect the most important input parameters which have a major influence on the model outputs. We applied a particular screening method, Morris screening, to identify important input parameters. Morris (1991) proposed the multiple elementary effects concept to compute two sensitivity measures. An elementary effect is defined as follows. Assume each model input parameter $X_i, i = 1, \ldots, k$ is divided into p levels in the space of the input parameters. An elementary effect EE_i is defined for each input parameter i as follows:

$$EE_i(\vec{x}) = f(x_1, \ldots, x_{i-1}, x_i + \Delta, \ldots, x_n) - f(\vec{x})/\Delta, \tag{1}$$

where Δ is a value in $\{1/(p-1), \ldots, 1-1/(p-1)\}$ and p is the number of levels. For each input factor, Morris provides two sensitivity measures μ and σ by sampling r elementary effects which are the mean and the standard deviation distribution of EE_i. The required number of simulations is equal to $r(k+1)$. In the computation of the μ value, EE_i values can be of opposite sign and cancel each other. For this reason, Campolongo et al. (2007) proposed the mean (μ^*) of the absolute elementary values instead of the μ value. Also, they suggested the choice of $r = [10, 20]$, $p = [4, 8]$, and $\Delta = p/2(p-1)$. In addition, we employed the mean (μ^*) of the absolute elementary values and $r = 20$, $p = 8$, and $\Delta = p/2(p-1)$.

3 Multilevel optimization framework based on the variable effect

In the most real-world problems, a large number of variables must be considered; but often shaping of the landscape is dominated by significant variables, i.e., variables with the most influence on the objective function. Most of the LSGO optimization algorithms do not discriminate among variables and consider equally all variables while some variables of an imbalanced LSGO problem have more effects on the fitness values. We propose a new optimization framework based on the most significant variables affecting on the objective function in which the problem can be decomposed into several low-dimensional levels of variables according to their effects. While CC algorithms focus mostly on decomposing an LSGO problem into nonseparable and separable subcomponents according interaction among variables, the proposed framework considers only imbalance feature, effects of variables on the objective function value, to increase the performance of LSGO algorithms. The main idea is that, the low-dimensional levels of decision variables with the most effect are identified and optimized at a few iterations. Then, all decision variables are optimized at remaining iterations with starting the fitter sub-solutions of important variables. By constructing the levels with the smaller decision variable space, the LSGO problem can transform into a problem with the smaller dimension but having higher contribution to the fitness value. In this section, we describe the details of MOFBVE algorithm. Let us start with the simplest version of the MOFBVE, Bilevel, which describes a model composed of three sequential optimization stages with two different search spaces. The steps of Bilevel optimization framework are shown in the Algorithm 1. The proposed Bilevel optimization framework has two different levels of search space, the important variables and all variables. It divides the optimization procedure into three stages: (1) the first stage optimizes LSGO problems as a whole at a few iterations (line 19); (2) the second stage optimizes only the important variables of LSGO problems at a few iterations (line 26), (3) and the last stage optimizes LSGO problems as a whole at remaining iterations (line 32). Every optimizer can be used as the subcomponent optimization algorithm in the both levels. We apply a self-adaptive DE with the neighborhood search algorithm (namely SaNSDE) (Yang et al. 2008c) in all stages, as the parent algorithm.

The sensitivity analysis method, called Morris method, is used to construct low-dimensional levels of variables according to their effects on the fitness value. First, Morris method is used to compute the main effect of each variable by two sensitivity measures μ^* and σ (line 6). Then, the k-means clustering method ($k = 2$) (MacQueen et al. 1967) is employed to cluster the variables of LSGO problems with two obtained features $\mu^\star$ and σ with Morris method (line 7). The variables with the similar effects on the fitness value are clustered into the same group by using the k-means algorithm. After that, the contribution of each group s is computed as follows:

Algorithm 1 :(best,best_val) $\Leftarrow$ **Bilevel**($func$, n, $max\_evaluate$)

1: //$func$, n, and $max\_evaluate$ are optimization function, the //dimension of the function, and the maximum number of fitness //evaluations, respectively.
2: $pop \Leftarrow$ **initialize**($popsize, n$)
3: //$popsize$ is the population size.
4: $max\_cycle = \frac{max\_evaluate}{popsize}$
5: //max_cycle is the maximum number of optimization cycles.
6: $(\mu, \sigma) \Leftarrow$ **Morris**(n) //Computing Morris sensitivity measures.
7: $(GO) \Leftarrow$ **K-means**($n, \mu, \sigma, 2$) //Clustering variables into 2 groups //according to two features μ and σ.
8: **for** $s = 1$ to 2 **do**
9: Calculate the main effect by Equation (2) for each subcomponent s
10: **end for**
11: Sort array GO, including 2 groups, in descending order according to value EF_s
12: $DK \Leftarrow length(GO[1])$ //Computing the size of the group 1 to //determine the number of iteration, i.e., $Iteropt$.
13: **if** $DK < 50$ **then**
14: $Iteropt = DK \times 20$
15: **else**
16: $Iteropt = DK \times 10$
17: **end if**
18: $(best, best\_val) \Leftarrow$ **evaluate**(pop)
19: $pop \Leftarrow$ **optimizer**($best$, pop, $Iteropt$)
20: $(best, best\_val) \Leftarrow$ **evaluate**(pop)
21: $i = Iteropt$
22: //Start optimizing the group 1 or the level 1
23: $index\_subpop \Leftarrow GO[1]$
24: $subpop \Leftarrow pop(:, index\_subpop)$
25: //the $subpop$ is a separate subpopulation of population including //the corresponding variables of the group 1
26: $subpop \Leftarrow$ **optimizer**($best$, $subpop$, $Iteropt$)
27: $pop(:, index\_subpop) \Leftarrow subpop$
28: $(best, best\_val) \Leftarrow$ **evaluate**(pop)
29: $i = i + 2 \times Iteropt$
30: //Start optimizing all variables at level 2
31: $Iteropt = max\_cycle - i$
32: $pop \Leftarrow$ **optimizer**($best$, pop, $Iteropt$)
33: $(best, best\_val) \Leftarrow$ **evaluate**(pop)

$$EF_s = \frac{\sum_{i=1}^{m} \mu_i^\star}{\sum_{i=1}^{n} \mu_i^\star}, \tag{2}$$

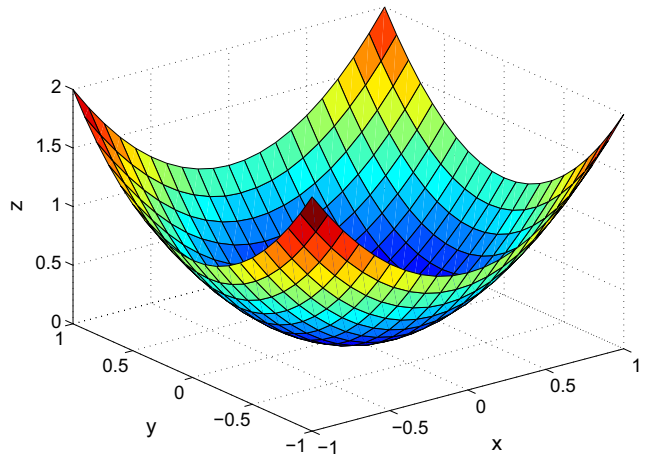
where parameters m and n are the number of variables in the group s and all variables of the function, respectively, and parameter $\mu_i^\star$ is Morris sensitivity measure $\mu^\star$ for the variable i. The measure EF_s can rank the importance of the various groups in terms of their influence on the fitness value. First, a low-dimensional level of the important variables is identified with the most contribution to the fitness function i.e., the group with maximum EF_s. In the first stage, a level of the problem search space with all decision variables is opti-

Algorithm 2 :(best,best_val) $\Leftarrow$ **MOFBVE**($func$, n, $max\_evaluate$, k)

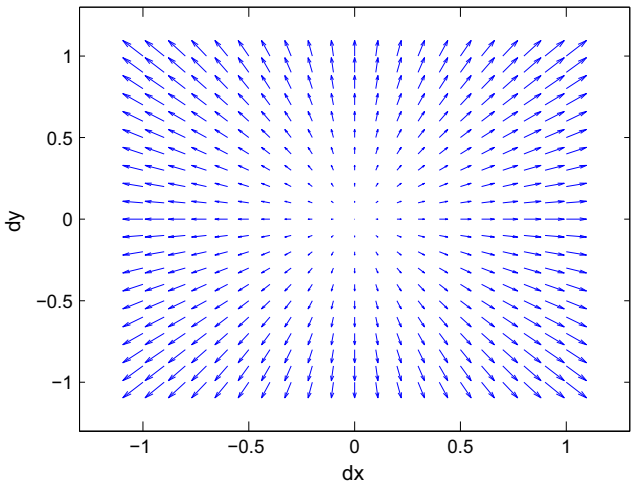
1: //$func$, n, $max\_evaluate$, and k are the optimization function, the //dimension of the function, the maximum number of fitness eval-//uations, and the number of levels in the MOFBVE algorithms, //respectively.
2: $pop \Leftarrow$ **initialize**($popsize, n$)
3: //$popsize$ is the population size.
4: $max\_cycle = \frac{max\_evaluate}{popsize}$
5: //max_cycle is the maximum number of optimization cycles.
6: $(\mu, \sigma) \Leftarrow$ **Morris**(n) //Computing Morris measures.
7: $(GO) \Leftarrow$ **K-means**(n, μ, σ, k) //Clustering variables into k groups //according to two features μ and σ.
8: **for** $s = 1$ to k **do**
9: Calculate the main effect by Equation (2) for each subcomponent s
10: **end for**
11: Sort array GO, including k groups, in descending order according to value EF_s
12: $(best, best\_val) \Leftarrow$ **evaluate**(pop)
13: $pop \Leftarrow$ **optimizer**($best$, pop, $Iteropt$)
14: $(best, best\_val) \Leftarrow$ **evaluate**(pop)
15: $i = Iteropt$ //Variable i is the total used iterations so far.
16: **for** $j = 1$ to $k - 1$ **do**
17: $DK \Leftarrow length(GO[j])$ Computing the size of the //group j to //determine the number of iteration, i.e., $Iteropt$.
18: $Iteropt = DK \times 10$
19: //Start optimizing the group j or the level j.
20: $index\_subpop \Leftarrow GO[j]$
21: $subpop \Leftarrow pop(:, index\_subpop)$
22: //the $subpop$ is a separate subpopulation of population including //the corresponding variables of the group j
23: $subpop \Leftarrow$ **optimizer**($best$, $subpop$, $Iteropt$)
24: $pop(:, index\_subpop) \Leftarrow subpop$
25: $(best, best\_val) \Leftarrow$ **evaluate**(pop)
26: **if** $j == k - 1$ **then**
27: $i = i + Iteropt$
28: **else**
29: $pop \Leftarrow$ **optimizer**($best$, pop, $Iteropt$)
30: $(best, best\_val) \Leftarrow$ **evaluate**(pop)
31: $i = i + 2 \times Iteropt$
32: **end if**
33: **end for**
34: //Start optimizing all variables at level k.
35: $Iteropt = max\_cycle - i$ //Computing remaining iterations.
36: $pop \Leftarrow$ **optimizer**($best$, pop, $Iteropt$)
37: $(best, best\_val) \Leftarrow$ **evaluate**(pop)

mized at a few determined iterations in the lines 18–20 (10 or $20 \times D_1$, where D_1 is the dimension of the most important group). If the dimension of the most important group is less than 50, then the number of optimization iterations is set to $20 \times D_1$; otherwise it is set to $10 \times D_1$ (lines 12–17). It should be noted that in the Bilevel algorithm, when the number of important variables is small (less than 50), they are optimized in more iterations. The reason for this is that the Bilevel algorithm uses the important variables in only one stage therefore it needs more number of optimization iterations for important variables to gain their effects on the optimization process especially when the number of variables becomes small. Then, the low-dimensional level of the important variables is optimized at the some iterations similar to previous stage (lines 23–28). The important feature of low-dimensional level is mapping an LSGO problem into a problem with a small number of variables but more influence on the fitness value. Therefore, the optimization algorithm with the search space including the most important group can obtain the efficient sub-solutions for the original problem. When this group is being optimized, all other variables are kept fixed. The best member of the first stage is employed to the other variables. Then, the obtained solutions from the low-dimensional level will be used as the initial population at the last stage. In the last stage (lines 31–33), the optimization algorithm with the original search space can start by the fitter candidate sub-solutions for more important variables which accelerate the convergence rate of the optimization algorithm at remaining iterations. Note that, if the low-dimensional level of the important variables are optimized at the first stage, then MOFBVE may converge to a local optimum. Therefore, in the first stage of MOFBVE, considering all variables reduces the possibility of population attraction to a local optimum.

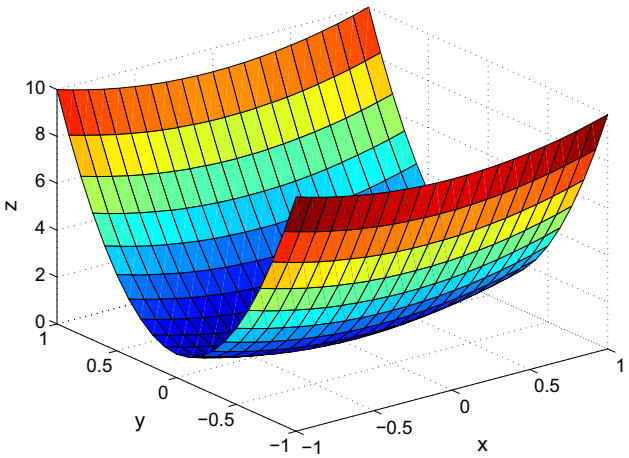
Now, we describe a generalized version of MOFBVE where the number of levels is more than two (i.e., $k > 2$) and its steps are shown in the Algorithm 2. First, the decision variables of an LSGO problem are clustered to k groups by using k-means clustering method with two sensitivity measures $\mu_i^\star$ and σ (lines 6–7). The measure EF_s is computed for all k groups (lines 8–10) and then they are sorted in the descending order according to the value EF_s (line 11). Then, the top $k-1$ groups are selected as $k-1$ levels of the search space and optimized at a sequence of some stages. Therefore, the k-MOFBVE includes the low-dimensional $k-1$ levels of the search space and a level of the problem search space with all decision variables. Also, an intra-stage, as the interconnection stage, is located between two stages of top low-dimensional levels and optimizes all variables (lines 29–30). In the first stage (lines 12–14), a level search space of the problem with all decision variables is optimized at a few determined iterations ($10 \times D_1$, where D_1 is the dimension of the most important group). In the second stage, the low-dimensional level of the important variables (in the group with maximum EF_s) is optimized at the some iterations similar to previous stage. Then, all variables are optimized at the some iterations similar to previous stage in the interconnection stage. In the next stage, the second group in the descending arrangement of groups is optimized at a few determined iterations ($10 \times D_2$, where D_2 is the dimension of the second group). This procedure continues in the similar way for all $k-1$ low-dimensional levels of the search space (lines 16–33). Finally, at level k, an optimization algorithm optimizes all variables at remaining iterations (lines 35–37). Note that since k-MOFBVE algorithm uses the most important variables with the different orders of effects in some its



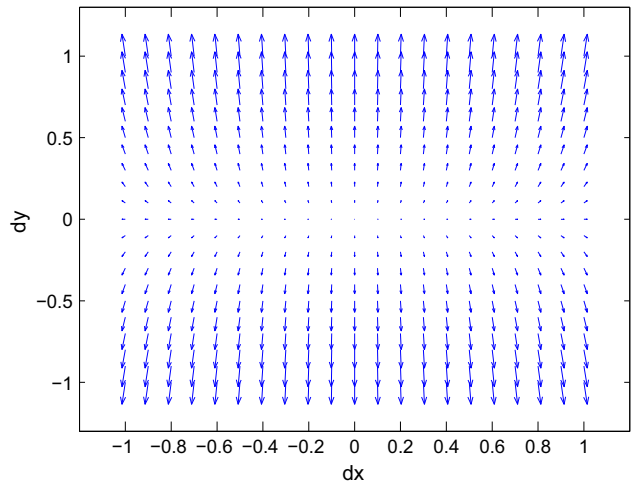
(a) Function $f(x,y) = x^2 + y^2$



(b) Gradient of function f



(c) Function $f_1(x,y) = x^2 + 16y^2$



(d) Gradient of function f_1

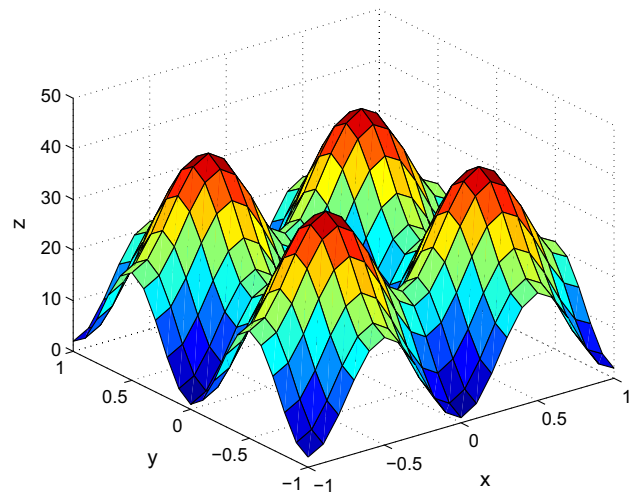
Fig. 1 The gradient of the functions f and f_1 for example 1

stages, it takes the benefit of important variables with $10 \times D_i$ iterations at each level i.

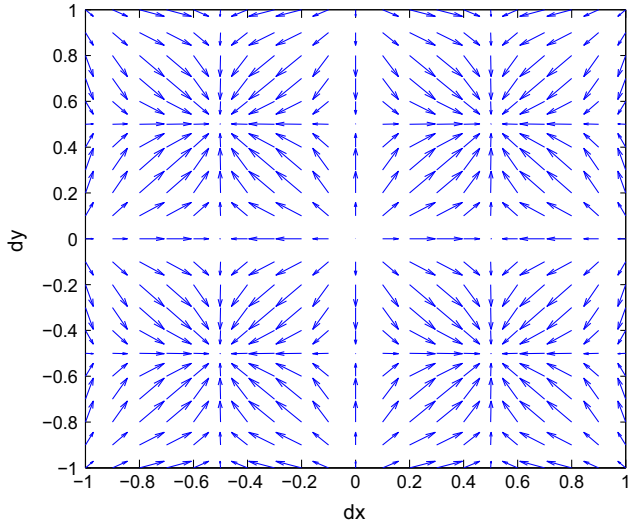
In general, each low-dimensional level i of the search space, the group ith in the descending arrangement of groups, is optimized at a few determined iterations in a stage (lines 20–23). While all other variables are held constant and then the obtained candidate solutions for ith group from the stage k will be used as the initial candidate solutions at the interconnection level. At interconnection level, all decision variables are optimized at a few iterations (line 29) which is equal to the number of the previous low-dimensional level iterations and then the obtained solutions will be used as the initial solutions for optimizing the next level $(i+1)$ of the search space at next stage. The interconnection level can switch between exploration in larger and smaller search spaces and is beneficial for exploring more promising regions around the obtained candidate sub-solutions of previous level. In MOFBVE, the search space of the optimization problem varies through the different levels according to ranking the importance of variables. The most important feature of this sequential mapping is the reduction of the search space to obtain the fitter candidate sub-solutions for groups including the most important variables with the different orders of effects. The last stage of MOFBVE takes the obtained solution of the previous stages as an initial population and improves them further.

4 Analyzing the imbalance feature impact on the optimization methods

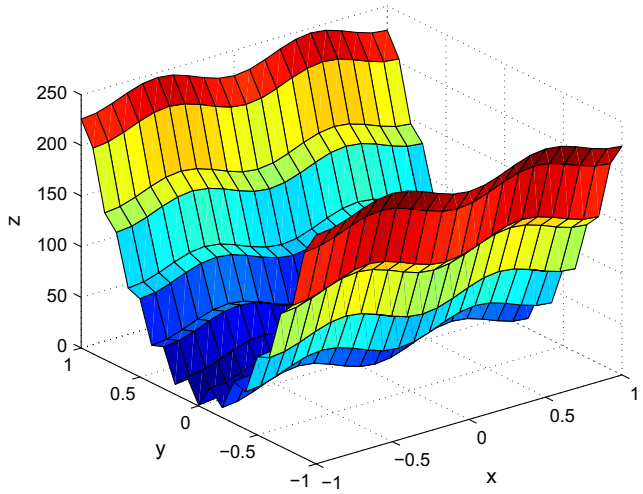
In this section, a theorem and some examples are presented in order to gain a better understanding of how considering the effect of variables can be beneficial for optimizing imbal-



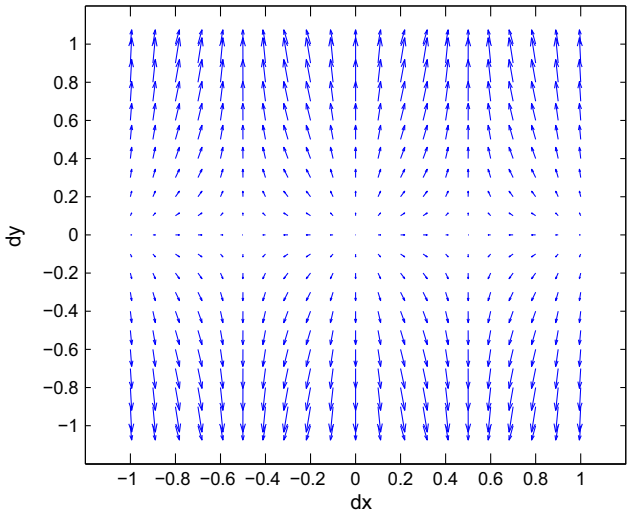
(a) Function $f(x, y) = x^2 - 10.\cos(2\pi x) + 10 + y^2 - 10.cos(2\pi y) + 10$



(b) Gradient of function f



(c) Function $f_1(x, y) = x^2 - 10.\cos(2\pi x) + 10 + 255y^2 - 10.cos(30\pi y) + 10$



(d) Gradient of function f_1

Fig. 2 The gradient of the functions f and f_1 for example 2

anced problems effectively. We first provide the following definitions based on the descriptions which are given in several studies (Li et al. 2013; Omidvar et al. 2014a; Doerr et al. 2013). A partially additively separable function is defined as follows (Li et al. 2013; Omidvar et al. 2014a):

$$F(\vec{x}) = \sum_{k=1}^{m} f_k(x_{v_k}), \quad v = [1, V], \tag{3}$$

where $\vec{x} = (x_1, x_2, \ldots, x_n) \in R^n$ is the decision variable vector, the function $F(x)$ is decomposed into m subcomponents where each subcomponent has V-dependent variables. A partially additively weighted separable function is defined as follows (Doerr et al. 2013):

$$F(\vec{x}) = \sum_{k=1}^{m} w_k . f_k(x_{v_k}), \quad v = [1, V] \tag{4}$$

$$\text{for} \quad 0 < w_1 < w_k < \cdots < w_m \tag{5}$$

This type of functions indicates the modular nature of the most real-world problems (Li et al. 2013; Omidvar et al. 2014a).

Also, we review the properties of the gradient vector in the optimization methods which is used to define following theorem. In the mathematics, the gradient is a vector-valued function whose components are the n partial derivatives of f. Many optimization methods concern the gradient vector to find and move towards the optimum of a problem. The signif-

Table 1 Results of Bilevel and MOFBVE with k (3, 4, and 5) on the modified CEC-2010 test functions

Function		3-MOFBVE	4-MOFBVE	5-MOFBVE	Bilevel
f_9	Mean	1.42e+11†	1.44e+11†	1.36e+11†	1.08e+11
	Std	2.68e+10	2.46e+10	2.62e+10	1.80e+10
f_{10}	Mean	1.04e+07‡	1.03e+07‡	1.11e+07$^{\approx}$	1.18e+07
	Std	2.02e+06	1.92e+06	1.33e+06	1.30e+06
f_{11}	Mean	2.66e+04$^{\approx}$	1.78e+04‡	2.68e+04†	2.57e+04
	Std	2.32e+04	1.45e+04	5.19e+04	8.83e+03
f_{12}	Mean	1.36e+06†	1.44e+06$^{\approx}$	5.26e+05‡	1.12e+06
	Std	5.38e+05	1.19e+06	1.72e+05	5.88e+05
f_{13}	Mean	1.11e+07‡	9.42e+06$^{\approx}$	1.20e+07$^{\approx}$	1.25e+07
	Std	5.91e+06	7.04e+06	8.24e+06	7.69e+06
f_{14}	Mean	1.64e+12†	1.68e+12†	1.63e+12†	1.40e+12
	Std	3.24e+11	3.60e+11	2.86e+11	2.23e+11
f_{15}	Mean	9.67e+07‡	9.04e+07‡	9.55e+07‡	1.02e+08
	Std	9.46e+06	1.08e+07	1.32e+07	1.12e+07
f_{16}	Mean	9.54e+05‡	5.77e+05‡	6.55e+05‡	1.28e+06
	Std	4.71e+05	4.16e+05	5.26e+05	2.51e+05
f_{17}	Mean	3.68e+07$^{\approx}$	3.52e+07$^{\approx}$	2.01e+07‡	2.84e+07
	Std	4.83e+07	5.71e+07	1.34e+07	1.60e+07
f_{18}	Mean	2.36e+08‡	5.57e+08†	1.49e+08‡	3.00e+08
	Std	1.87e+08	1.97e+09	6.36e+07	1.91e+08
	w/t/l	5/2/3	4/3/3	5/2/3	–

icant properties of the gradient vector (Arora 2004; Rao and Rao 2009) are as follows: (1) the gradient vector $\vec{c}$ of a function $f(x_1, x_2, \ldots, x_n)$ at the point $x^* = (x_1{}^*, x_2{}^*, \ldots, x_n{}^*)$ is orthogonal to the tangent hyperplane for the surface $f(x_1, x_2, \ldots, x_n) = constant$; (2) the gradient shows a direction of maximum rate of increase for the function $f(x)$ at the point x^*; (3) the maximum rate of change of $F(\vec{x})$ at any point x^* is the magnitude of the gradient vector. These properties show that the gradient vector at any point x^* represents a direction of maximum increase in the function $f(x)$ and the rate of increase is the magnitude of the vector.

Theorem 1 *If $F(\vec{x})$ is partially additively weighted separable function with the dimension $n \geq m \geq 2$ and m subcomponents, then the subcomponents with the more effects on the objective function have more impacts than other subcomponents on the direction of steepest ascent and maximum change rate of the function.*

Proof As mentioned above, the gradient vector indicates the direction of steepest ascent and the maximum rate of function change which is the magnitude of the gradient vector. Therefore, we first prove that the subcomponents with the more effects have more impacts on the magnitude and direction of the gradient. The gradient of the function $F(\vec{x})$ is computed as follows:

$$F(\vec{x}) = \sum_{k=1}^{m} w_k.f_k(x_{v_k}), \quad \text{where } x_{v_i} = (x_{si}, \ldots, x_{li}) \tag{6}$$

$$\nabla F = w_1.\nabla f_1 + w_2.\nabla f_2 + \cdots + w_m.\nabla f_m \tag{7}$$

$$= w_1.\sum_{z=s1}^{l1} \frac{\partial f_1}{\partial x_z}.\vec{e}_z + w_2.\sum_{z=s2}^{l2} \frac{\partial f_2}{\partial x_z}.\vec{e}_z + \cdots \tag{8}$$

$$+ w_m.\sum_{z=sm}^{lm} \frac{\partial f_m}{\partial x_z}.\vec{e}_z \tag{9}$$

$$= \sum_{k=1}^{m} w_k.\sum_{z=sb}^{lb} \frac{\partial f_k}{\partial x_z}.\vec{e}_z, \quad \text{where } e_{zi} = \begin{cases} 1 & \forall i = z \\ 0 & \text{otherwise} \end{cases}, \tag{10}$$

$$\text{and therefore} ||\nabla F|| = \sqrt{\sum_{b=1}^{k} \left(w_b.\sum_{z=sb}^{lb} \frac{\partial f_b}{\partial x_z} \right)^2} \tag{11}$$

and the subcomponents have the various effects and $0 < w_1 < w_k < \cdots < w_m$ thus the partial derivatives of the gradient vector can be sorted as follows:

$$w_1.\sum_{z=s1}^{l1} \left| \frac{\partial f_1}{\partial x_z} \right| < w_2.\sum_{z=s2}^{l2} \left| \frac{\partial f_2}{\partial x_z} \right| < \cdots < w_m.\sum_{z=sm}^{lm} \left| \frac{\partial f_m}{\partial x_z} \right| \tag{12}$$

□

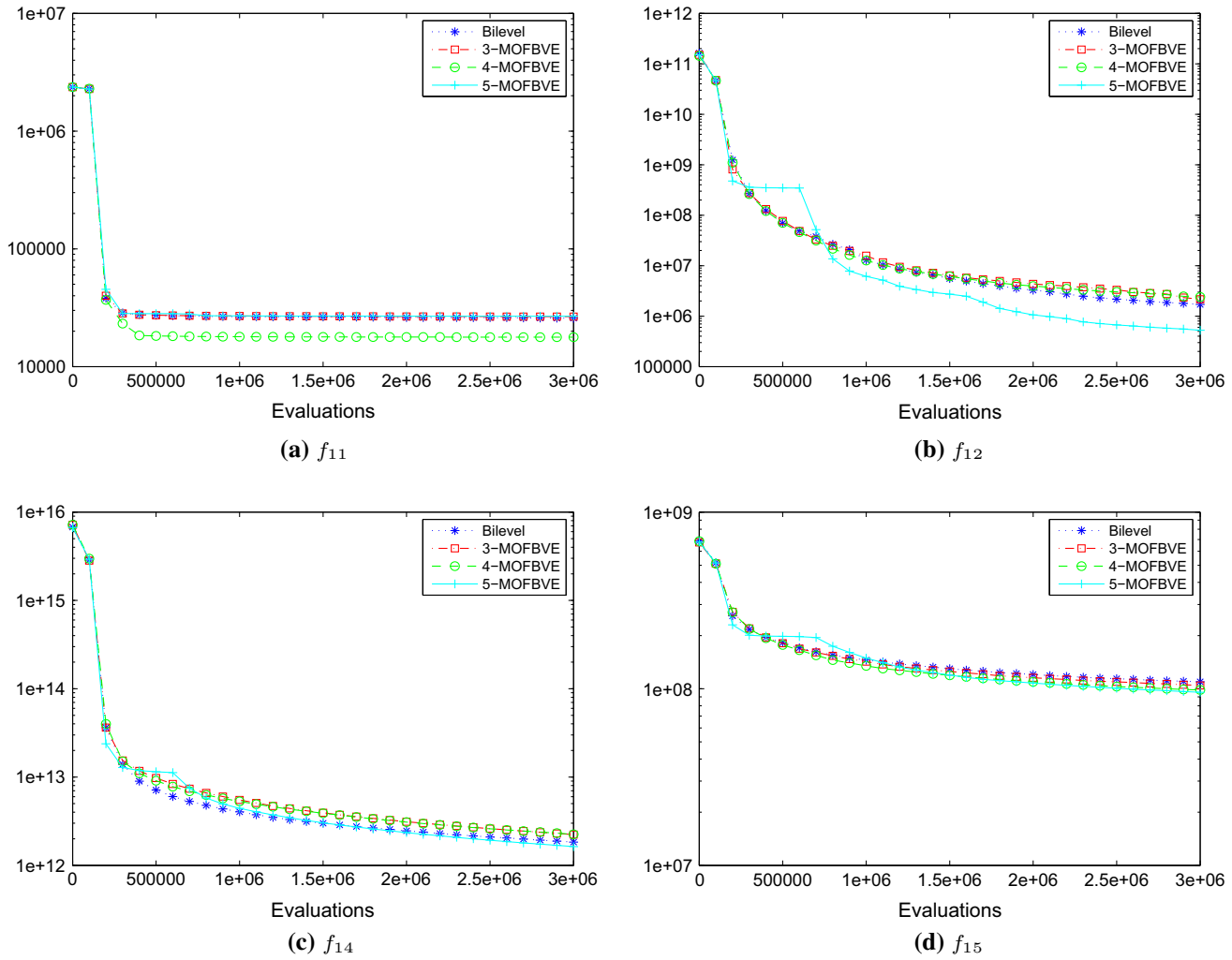


Fig. 3 Convergence plots of f_{11}, f_{12}, f_{14}, and f_{15} of the modified CEC-2010 test functions. The results of the Bilevel framework and k-MOFBVE algorithms were averaged over 25 runs. The vertical axis is the function value and the horizontal axis is the number of function evaluations

Hence, the important subcomponents have more effects on the magnitude and direction of the gradient vector. Therefore, it is reasonable that by focusing on important variable, an optimization algorithm can find the promising regions, because the search space is directed to move more toward the direction of steepest ascent. MOFBVE attempts to identify the most important variables with the Morris method and then by mapping an LSGO problem into a smaller size problem, it spends more iteration for variables with higher contribution to find promising solutions.

Note that theorem 1 has been proved for a partially additively weighted separable function with the dimension $n \geq 2$, but two examples are presented in two dimensions since two-dimensional functions and their gradients can be visualized better than higher dimensions.

Example 1 Consider a function $f(x, y) = x^2 + y^2$ with two variables which have the equal effects. First, $\mu_i^\star$ values of two variables are computed by the Morris method. Then, the values of EF_s are computed by Morris method as follows:

$$EF_1 = \frac{\mu_1^\star}{\sum_{i=1}^{2} \mu_i^\star} = \frac{1.01}{2.02} = 0.5 \tag{13}$$

$$EF_2 = \frac{\mu_2^\star}{\sum_{i=1}^{2} \mu_i^\star} = \frac{1.01}{2.02} = 0.5 \tag{14}$$

Therefore, the Morris method identifies that two variables have the equal effects. For defining a imbalanced function, we change the above function as follows:

$$f_1(x, y) = x^2 + 16y^2 \tag{15}$$

this function has two variables with the unequal effects. First, $\mu_i^\star$ values of two variables are computed by the Morris method. Then, the values of EF_s are computed by Morris method as follows:

Table 2 Results of Bilevel-I and MOFBVE-I with k (3, 4, and 5) algorithms on the modified CEC-2010 test functions

Function		3-MOFBVE-I	4-MOFBVE-I	5-MOFBVE-I	Bilevel-I
f_9	Mean	1.20e+11$^{\approx}$	1.11e+10$^{\approx}$	1.25e+10$^{\approx}$	1.22e+11
	Std	2.29e+10	2.43e+10	2.76e+10	3.13e+10
f_{10}	Mean	1.12e+07$^{\approx}$	1.06e+07$^{\approx}$	1.20e+07$^{\approx}$	1.14e+07
	Std	1.75e+06	1.50e+06	1.57e+06	1.51e+06
f_{11}	Mean	1.64e+04‡	1.24e+04‡	1.25e+04‡	1.51e+05
	Std	1.17e+04	1.10e+04	9.36e+03	9.03e+04
f_{12}	Mean	1.11e+06‡	6.83e+05‡	9.39e+05‡	1.56e+06
	Std	1.09e+06	2.85e+05	3.38e+05	7.72e+05
f_{13}	Mean	8.27e+06‡	1.06e+07‡	1.01e+07‡	1.31e+07
	Std	3.78e+06	3.62e+06	3.85e+06	4.05e+06
f_{14}	Mean	1.52e+12$^{\approx}$	1.59e+12$^{\approx}$	1.40e+012$^{\approx}$	1.51e+12
	Std	3.24e+11	3.55e+011	2.69e+011	2.97e+11
f_{15}	Mean	1.01e+08$^{\approx}$	9.72e+07$^{\approx}$	1.01e+08$^{\approx}$	1.01e+08
	Std	1.25e+07	1.06e+07	1.14e+07	8.84e+06
f_{16}	Mean	5.57e+05‡	4.70e+05‡	7.75e+05‡	2.32e+06
	Std	3.22e+05	2.11e+05	4.00e+05	9.38e+05
f_{17}	Mean	2.55e+07$^{\approx}$	1.61e+07‡	2.09e+07‡	4.18e+07
	Std	1.80e+07	7.06e+06	1.47e+07	6.40e+07
f_{18}	Mean	1.53e+08$^{\approx}$	1.39e+08$^{\approx}$	3.68e+08$^{\approx}$	2.67e+08
	Std	8.14e+07	1.05e+08	7.06e+08	3.39e+08
	w/t/l	4/6/0	5/5/0	5/5/0	–

Symbols '†', '‡', and '≈' denote the compared algorithms are worse than, better than, or similar to Bilevel-I, respectively

Table 3 Results of Bilevel and MOFBVE with k (3, 4, and 5) on the modified normal CEC-2010 test functions

Function		3-MOFBVE	4-MOFBVE	5-MOFBVE	Bilevel
f_9	Mean	3.18e+10$^{\approx}$	3.31e+10‡	2.85e+10$^{\approx}$	2.57e+10
	Std	8.86e+09	8.20e+09	8.12e+09	6.68e+09
f_{10}	Mean	2.06e+05$^{\approx}$	1.96e+05$^{\approx}$	1.97e+05$^{\approx}$	2.05e+05
	Std	2.43e+04	2.37e+04	3.09e+04	2.68e+04
f_{11}	Mean	1.54e+03$^{\approx}$	1.36e+03‡	1.06e+03‡	1.84e+03
	Std	7.25e+02	5.88e+02	4.48e+02	5.67e+02
f_{12}	Mean	1.30e+06$^{\approx}$	1.20e+06$^{\approx}$	6.89e+05‡	9.50e+05
	Std	9.91e+05	6.69e+05	4.36e+05	2.91e+05
f_{13}	Mean	1.45e+07$^{\approx}$	1.25e+07‡	1.12e+07‡	1.99e+07
	Std	1.10e+07	8.05e+06	9.12e+06	1.04e+07
f_{14}	Mean	3.46e+12$^{\approx}$	3.55e+12$^{\approx}$	3.56e+12$^{\approx}$	3.36e+12
	Std	7.97e+11	7.82e+11	8.13e+11	9.65e+11
f_{15}	Mean	3.06e+07$^{\approx}$	2.86e+07‡	3.02e+07‡	3.25e+07
	Std	3.98e+06	4.43e+06	2.78e+06	4.26e+06
f_{16}	Mean	1.95e+05$^{\approx}$	1.26e+05‡	1.76e+05‡	2.28e+05
	Std	1.94e+05	8.87e+04	3.30e+05	9.74e+04
f_{17}	Mean	8.61e+07$^{\approx}$	8.14e+07†	1.60e+07‡	4.07e+07
	Std	1.26e+08	9.97e+07	1.60e+07	5.70e+07
f_{18}	Mean	2.40e+10$^{\approx}$	2.34e+10$^{\approx}$	2.50e+10$^{\approx}$	3.24e+10
	Std	1.62e+10	9.92e+09	1.10e+10	2.49e+10
	w/t/l	0/10/0	5/4/1	6/4/0	–

Symbols '‡', '†', and '≈' denote the compared algorithms are worse than, better than, or similar to Bilevel, respectively

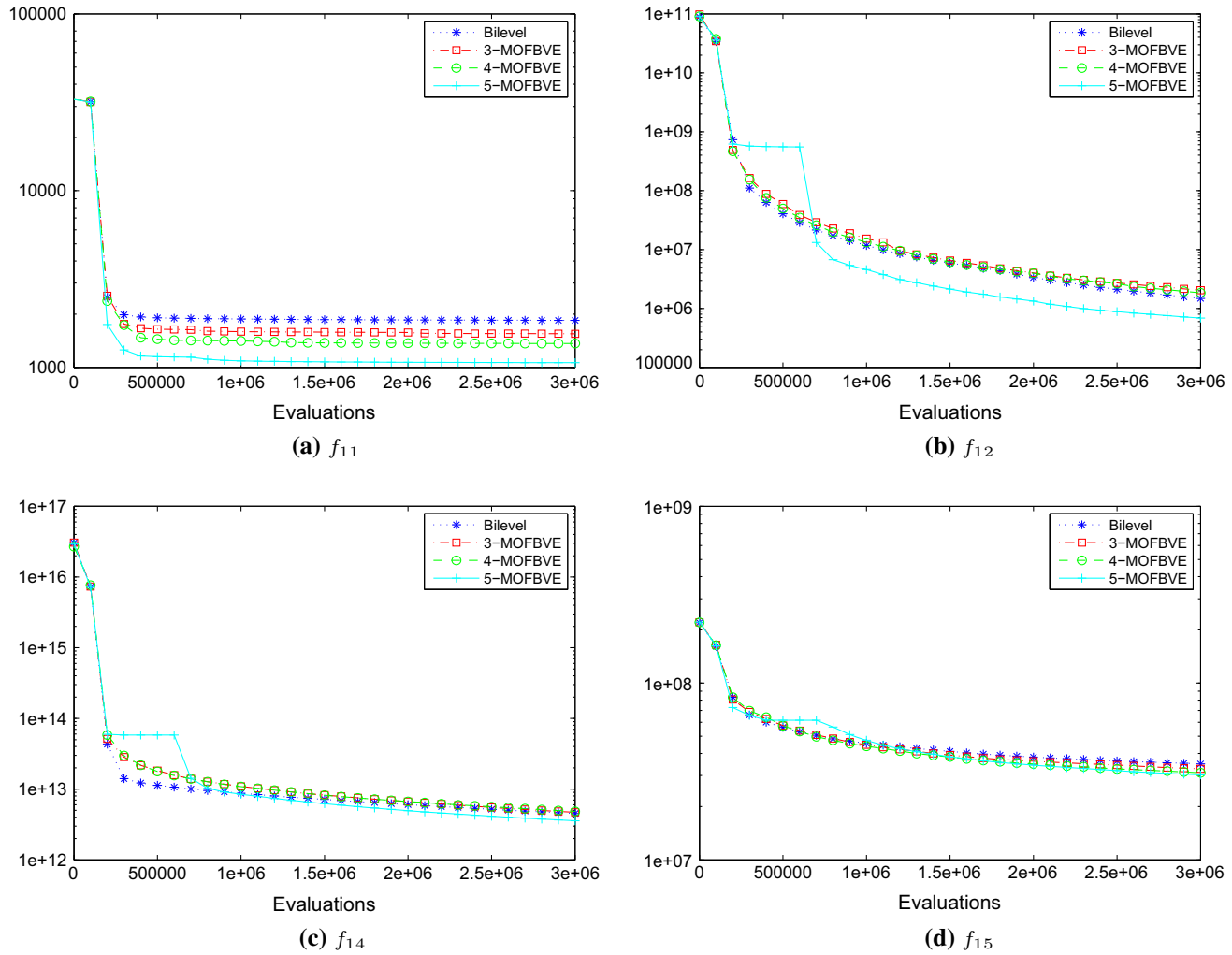


Fig. 4 Convergence plots of f_{11}, f_{12}, f_{14}, and f_{15} of the modified normal CEC-2010 test functions. The results of the Bilevel framework and k-MOFBVE algorithms were averaged over 25 runs. The vertical axis is the function value and the horizontal axis is the number of function evaluations

$$EF_1 = \frac{\mu_1^\star}{\sum_{i=1}^{2} \mu_i^\star} = \frac{0.99}{16.76} = 0.06 \tag{16}$$

$$EF_2 = \frac{\mu_2^\star}{\sum_{i=1}^{2} \mu_i^\star} = \frac{15.77}{16.76} = 0.94 \tag{17}$$

Therefore, the Morris method identifies the variable y as the important variable and the multilevel algorithm spends more optimization iterations to achieve better solutions. Figure 1 shows the gradient and function plots for two functions f_1 and f. The arrows in Fig. 1 show the magnitude and direction of the gradient. It can be seen from Fig. 1, the arrows for the function f are parallel to the diagonal lines $y = x$ and $y = -x$, therefore the impact of variables x and y is equal on the gradient vector. Also, the arrows for the function f_1 are approximately parallel to the y-axis thus the impact of variable y is more than variable x on the gradient vector and the direction of the gradient is about in the direction of the y-axis.

Example 2 Consider a function $f(x, y) = x^2 - 10.\cos(2\pi x) + 10 + y^2 - 10.cos(2\pi y) + 10$ with two variables which have the equal effects. First, $\mu_i^\star$ values of two variables are computed by the Morris method. Then, the values of EF_s are computed by Morris method as follows:

$$EF_1 = \frac{\mu_1^\star}{\sum_{i=1}^{2} \mu_i^\star} = \frac{23.65}{47.58} = 0.5 \tag{18}$$

$$EF_2 = \frac{\mu_2^\star}{\sum_{i=1}^{2} \mu_i^\star} = \frac{23.93}{47.58} = 0.5 \tag{19}$$

Therefore, the Morris method identifies that two variables have the equal effects. For defining a imbalanced function, we change the above function as follows:

$$f_1(x, y) = x^2 - 10.\cos(2\pi x) + 10 + 255y^2 - 10.cos(30\pi y) + 10 \tag{20}$$

Table 4 Results of Bilevel-I and MOFBVE-I with k (3, 4, and 5) on the modified normal CEC-2010 test functions

Function		3-MOFBVE-I	4-MOFBVE-I	5-MOFBVE-I	Bilevel-I
f_9	Mean	2.68e+10$^{\approx}$	2.50e+10$^{\approx}$	2.57e+10$^{\approx}$	2.66e+10
	Std	2.68e+10	6.91e+09	7.29e+09	8.57e+09
f_{10}	Mean	2.08e+05$^{\approx}$	2.12e+05$^{\approx}$	2.08e+05$^{\approx}$	2.29e+05
	Std	2.08e+05	2.60e+04	2.20e+04	7.40e+04
f_{11}	Mean	9.57e+02‡	9.71e+02‡	1.30e+03$^{\approx}$	1.22e+03
	Std	9.57e+02	1.28e+02	4.64e+02	2.30e+02
f_{12}	Mean	9.05e+05‡	7.19e+05‡	1.02e+06‡	1.89e+06
	Std	9.05e+05	3.14e+05	4.31e+05	1.99e+06
f_{13}	Mean	1.77e+07$^{\approx}$	2.10e+07$^{\approx}$	2.03e+07$^{\approx}$	1.95e+07
	Std	1.77e+07	7.91e+06	7.66e+06	1.05e+07
f_{14}	Mean	4.00e+12$^{\approx}$	3.49e+12$^{\approx}$	3.48e+12$^{\approx}$	3.34e+12
	Std	4.00e+12	9.99e+11	1.04e+12	9.34e+11
f_{15}	Mean	2.88e+07‡	2.73e+07‡	3.11e+07‡	3.51e+07
	Std	2.88e+07	3.86e+06	4.01e+06	5.25e+06
f_{16}	Mean	1.88e+05$^{\approx}$	1.12e+05$^{\approx}$	1.05e+05‡	1.57e+05
	Std	1.88e+05	8.30e+04	1.04e+05	8.71e+04
f_{17}	Mean	2.30e+07‡	1.66e+07‡	3.35e+07‡	7.24e+07
	Std	2.30e+07	1.22e+07	3.64e+07	6.88e+07
f_{18}	Mean	2.27e+10$^{\approx}$	2.54e+10$^{\approx}$	2.85e+10$^{\approx}$	2.80e+10
	Std	2.27e+10	9.29e+09	1.91e+10	1.31e+10
	w/t/l	4/6/0	4/6/0	4/6/0	–

Symbols '†', '‡', and '≈' denote the compared algorithms are worse than, better than, or similar to Bilevel, respectively

This function has two variables with the unequal effects. First, $\mu_i^\star$ values of two variables are computed by the Morris method. Then, the values of EF_s are computed by Morris method as follows:

$$EF_1 = \frac{\mu_1^\star}{\sum_{i=1}^{2} \mu_i^\star} = \frac{23.53}{241.08} = 0.1 \tag{21}$$

$$EF_2 = \frac{\mu_2^\star}{\sum_{i=1}^{2} \mu_i^\star} = \frac{217.55}{241.08} = 0.9 \tag{22}$$

Therefore, the Morris method identifies the variable y as the important variable and the multilevel algorithm spends more optimization iterations to achieve better solutions. Figure 2 shows the gradient and function plots for two function f_1 and f. The arrows in Fig. 2 show the magnitude and direction of the gradient. It can be seen in Fig. 2, the arrows for the function f are parallel to the diagonal lines $y = x$ and $y = -x$, therefore the impact of variables x and y is equal on the gradient vector. Also, the arrows for the function f_1 are approximately parallel to the y-axis, thus the impact of variable y is more than variable x on the gradient vector and the direction of the gradient is in the direction of the y-axis.

5 Experimental studies and discussion

5.1 Experiment setup

In order to evaluate the performance of proposed MOFBVE algorithm, we have utilized the same set of modified CEC-2010 benchmark functions which are used in the LSGO field by Chen et al. (2010); Omidvar et al. (2014a); Mahdavi et al. (2014); Zhao et al. (2011); Wang et al. (2013); Molina et al. (2010) and also a new set of benchmark functions based on the CEC-2010 benchmark functions. The CEC-2010 benchmark functions (Appendix B) were provided by the CEC-2010 Special Session and Competition on LSGO (Tang et al. 2010). In this benchmark test set, there are five types of functions as follows (Appendix B):

- C1: Separable functions (f_1–f_3)
- C2: Single-group m-nonseparable functions (f_4–f_8)
- C3: $\frac{n}{2m}$-group m-nonseparable functions (f_9–f_{13})
- C4: $\frac{n}{m}$-group m-nonseparable functions (f_{14}–f_{18})
- C5: Nonseparable functions (f_{19}–f_{20}),

where n is the dimension of the function and m is the number of variables in each nonseparable subcomponent. In the modified CEC-2010 benchmark functions, Omidvar et al. (2014a)

Table 5 Results of Bilevel and MOFBVE with k (3, 4, and 5) on the CEC-2013 LSGO benchmark test functions

Function		3-MOFBVE	4-MOFBVE	5-MOFBVE	Bilevel
f_4	Mean	9.92e+09$^{\approx}$	9.13e+09$^{\approx}$	7.29e+09‡	9.09e+09
	Std	5.01e+09	3.08e+09	3.63e+09	2.60e+09
f_5	Mean	2.77e+06$^{\approx}$	2.80e+06$^{\approx}$	3.01e+06$^{\approx}$	2.69e+06
	Std	4.19e+05	3.61e+05	5.43e+05	5.30e+05
f_6	Mean	9.33e+04$^{\approx}$	9.25e+04$^{\approx}$	8.81e+04$^{\approx}$	8.56e+04
	Std	2.91e+04	2.15e+04	2.60e+04	2.41e+04
f_7	Mean	5.85e+06$^{\approx}$	9.35e+06$^{\approx}$	6.53e+06$^{\approx}$	5.86e+06
	Std	2.21e+06	1.26e+07	2.80e+06	2.18e+06
f_8	Mean	1.81e+13‡	2.54e+13$^{\approx}$	2.88e+13$^{\approx}$	2.31e+13
	Std	1.37e+13	1.53e+13	1.78e+13	1.13e+13
f_9	Mean	2.65e+08$^{\approx}$	2.62e+08$^{\approx}$	1.94e+08‡	2.81e+08
	Std	3.16e+07	2.62e+07	3.30e+07	3.09e+07
f_{10}	Mean	2.09e+04‡	2.55e+03‡	1.89e+03‡	3.34e+04
	Std	2.22e+04	3.31e+02	1.15e+03	2.17e+04
f_{11}	Mean	4.55e+08$^{\approx}$	4.48e+08$^{\approx}$	3.83e+09$^{\approx}$	7.64e+08
	Std	3.66e+08	3.97e+08	1.34e+10	1.04e+09
f_{13}	Mean	5.31e+08$^{\approx}$	5.68e+08$^{\approx}$	6.56e+08$^{\approx}$	5.27e+08
	Std	1.95e+08	2.195e+08	2.80e+08	1.48e+08
f_{14}	Mean	2.15e+09$^{\approx}$	7.57e+08$^{\approx}$	6.46e+08$^{\approx}$	6.90e+08
	Std	8.52e+09	1.53e+09	7.15e+08	1.02e+09
	w/t/l	2/8/0	1/9/0	3/7/0	–

Symbols '†', '‡', and '≈' denote the compared algorithms are worse than, better than, or similar to Bilevel, respectively

Table 6 Results of Bilevel and MOFBVE with k (3, 4, and 5) on the CEC-2013 LSGO benchmark test functions

Function		3-MOFBVE-I	4-MOFBVE-I	5-MOFBVE-I	Bilevel-I
f_4	Mean	7.14e+09$^{\approx}$	5.55e+09$^{\approx}$	5.27e+09$^{\approx}$	5.92e+09
	Std	3.58e+09	2.68e+09	2.41e+09	2.14e+09
f_5	Mean	2.88e+06$^{\approx}$	2.96e+06$^{\approx}$	3.08e+06$^{\approx}$	3.15e+06
	Std	5.50e+05	4.76e+05	7.52e+05	4.87e+05
f_6	Mean	7.77e+04$^{\approx}$	8.56e+04$^{\approx}$	8.61e+04$^{\approx}$	9.16e+04
	Std	3.13e+04	2.10e+04	1.60e+04	1.63e+04
f_7	Mean	5.00e+06$^{\approx}$	3.95e+06‡	3.54e+06‡	4.30e+06
	Std	2.75e+06	2.10e+06	9.68e+05	1.08e+06
f_8	Mean	1.66e+13$^{\approx}$	2.15e+13$^{\approx}$	1.99e+13$^{\approx}$	1.94e+13
	Std	6.38e+12	1.28e+13	8.26e+12	7.84e+12
f_9	Mean	2.17e+08‡	2.04e+08‡	2.02e+08‡	2.69e+08
	Std	3.10e+07	3.33e+07	3.13e+07	2.70e+07
f_{10}	Mean	4.32e+03‡	2.35e+03‡	3.42e+03‡	4.26e+04
	Std	9.05e+03	4.76e+02	9.19e+03	1.51e+04
f_{11}	Mean	2.75e+08$^{\approx}$	2.26e+08‡	2.20e+08‡	3.11e+08
	Std	8.47e+07	1.05e+08	6.67e+07	9.39e+07
f_{13}	Mean	5.01e+08$^{\approx}$	4.72e+08$^{\approx}$	5.16e+08$^{\approx}$	5.15e+08
	Std	2.09e+08	2.45e+08	1.61e+08	1.86e+08
f_{14}	Mean	2.81e+08‡	5.42e+08$^{\approx}$	5.37e+08$^{\approx}$	9.67e+08
	Std	2.45e+08	4.64e+08	7.43e+08	1.51e+09
	w/t/l	3/7/0	4/6/0	4/6/0	–

Symbols '†', '‡', and '≈' denote the compared algorithms are worse than, better than, or similar to Bilevel, respectively

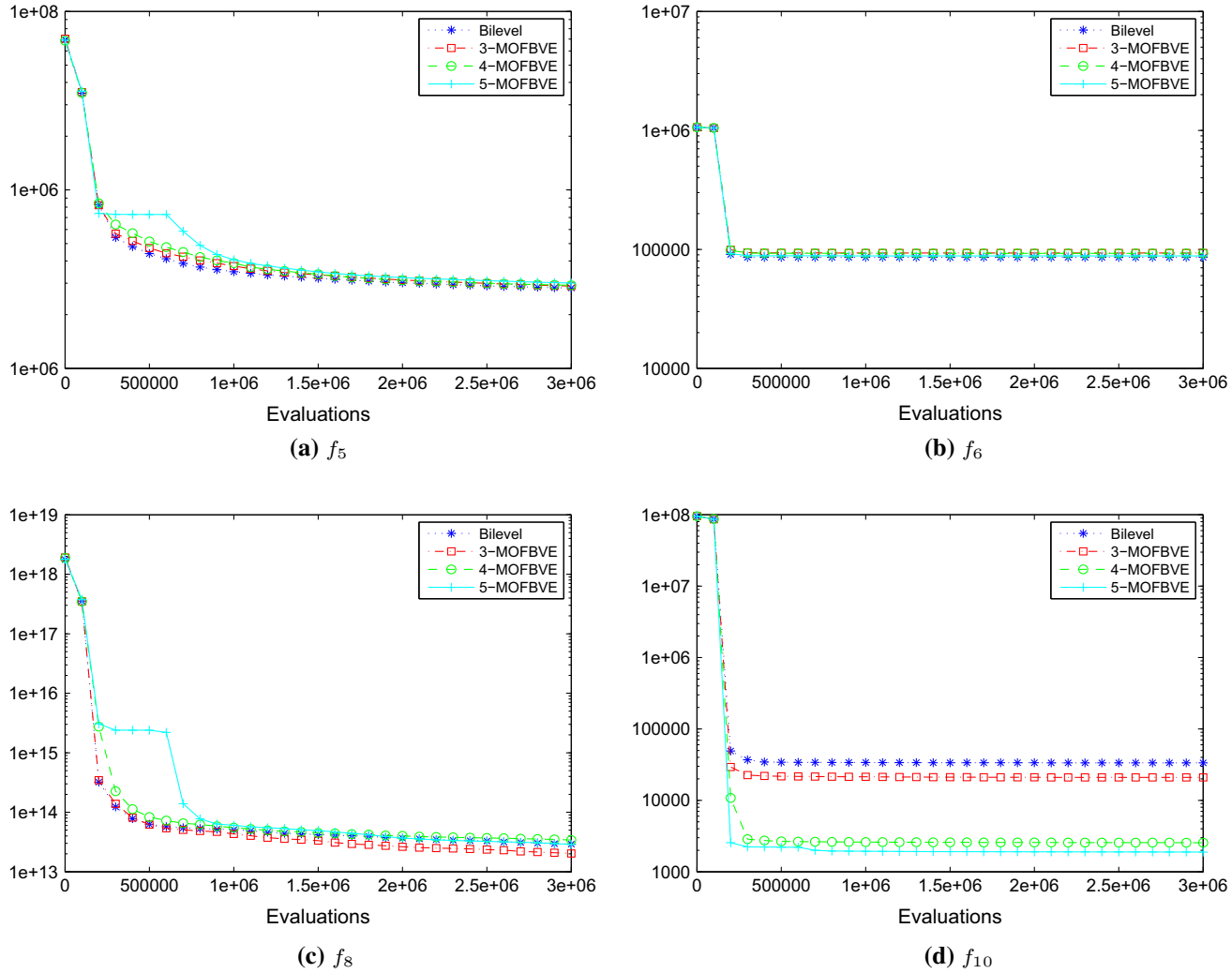


Fig. 5 Convergence plots of f_5, f_6, f_8, and f_{10} of the CEC-2013 LSGO benchmark functions. The results of the Bilevel framework and k-MOFBVE algorithms were averaged over 25 runs. The vertical axis is the function value and the horizontal axis is the number of function evaluations

created an imbalance effect by multiplying some coefficients to subcomponents of the CEC-2010 benchmark functions (f_9–f_{18}). Also, we generated a new set of benchmark functions, the modified normal CEC-2010 test functions, similar to the modified CEC-2010 test functions while a component is multiplied by a normal distribution coefficient to make an imbalance effect. The normal coefficient for ith nonseparable subcomponent is calculated as following formula proposed in Li et al. (2013):

$$C_i = 10^{3N(0,1)} \tag{23}$$

The normal coefficients are provided in the Appendix A.

In the modified CEC-2010 test functions, the coefficients of subcomponents are the same for all functions while the normal coefficients can generate the different imbalance effects of subcomponents for each function in the modified normal CEC-2010 test functions. MOFBVE algorithms with 3, 4, and 5 levels use two or more groups of important variables; therefore they can perform on function with two or more imbalanced nonseparable subcomponents. So, we selected ten functions (f_9–f_{18}) which have two or more imbalanced nonseparable subcomponents from the modified normal and modified CEC-2010 benchmark functions in all our experiments.

In addition, some experiments are conducted on the CEC-2013 LSGO benchmark functions (Li et al. 2013) (on imbalanced nonseparable functions, i.e., f_4–f_{11} and f_{13}–f_{14}) with new transformations such as ill-conditioning, symmetry breaking, irregularities, and having subcomponents with non-uniform subcomponent sizes. The CEC-2013 LSGO benchmark functions are divided into five classes: fully separable functions (C1: f_1–f_3), partially separable functions with a separable subcomponent (C2: f_4–f_7), partially separable functions with no separable subcomponents (C3: f_8–f_{11}), overlapping functions (C4: f_{12}–f_{14}) and one

Table 7 Results of Bilevel and random Bilevel on the modified CEC-2010 test functions

Function		Random Bilevel	Bilevel
f_9	Mean	1.31e+01‡	1.08e+11
	Std	2.40e+10	1.80e+10
f_{10}	Mean	1.23e+07$^{\approx}$	1.18e+07
	Std	1.42e+06	1.30e+06
f_{11}	Mean	1.34e+05‡	2.57e+04
	Std	9.76e+04	8.83e+03
f_{12}	Mean	1.45e+06‡	1.12e+06
	Std	5.42e+05	5.88e+05
f_{13}	Mean	1.40e+07$^{\approx}$	1.25e+07
	Std	5.54e+06	7.69e+06
f_{14}	Mean	1.60e+12‡	1.40e+12
	Std	2.95e+11	2.23e+11
f_{15}	Mean	1.04e+08$^{\approx}$	1.02e+08
	Std	1.0e+07	1.12e+07
f_{16}	Mean	1.77e+06‡	1.28e+06
	Std	9.23e+05	2.51e+05
f_{17}	Mean	4.64e+07$^{\approx}$	2.84e+07
	Std	4.35e+07	1.60e+07
f_{18}	Mean	9.36e+08$^{\approx}$	3.00e+08
	Std	3.02e+09	1.91e+08
	w/t/l	5/5/0	–

Symbols '†', '‡', and '≈' denote Bilevel is worse than, better than, or similar to random Bilevel, respectively

Table 8 Results of Bilevel and random Bilevel on the modified normal CEC-2010 test functions

Function		Random Bilevel	Bilevel
f_9	Mean	2.54e+10$^{\approx}$	2.57e+10
	Std	7.21e+09	6.68e+09
f_{10}	Mean	2.17e+05‡	2.05e+05
	Std	2.25e+04	2.68e+04
f_{11}	Mean	3.04e+03‡	1.84e+03
	Std	5.89e+02	5.67e+02
f_{12}	Mean	1.32e+06‡	9.50e+05
	Std	8.40e+05	2.91e+05
f_{13}	Mean	2.12e+07$^{\approx}$	1.99e+07
	Std	7.91e+06	1.04e+07
f_{14}	Mean	3.08e+12$^{\approx}$	3.36e+12
	Std	6.76e+11	9.65e+11
f_{15}	Mean	3.28e+07$^{\approx}$	3.25e+07
	Std	3.06e+06	4.26e+06
f_{16}	Mean	1.22e+06‡	2.28e+05
	Std	9.23e+05	9.74e+04
f_{17}	Mean	8.84e+07‡	4.07e+07
	Std	1.61e+08	5.70e+07
f_{18}	Mean	2.44e+10$^{\approx}$	3.24e+10
	Std	7.74e+09	2.49e+10
	w/t/l	5/5/0	–

Symbols '‡', '†', and '≈' denote Bilevel is worse than, better than, or similar to random Bilevel, respectively

fully nonseparable function (C5:f_{15}). The sizes of all nonseparable subcomponents in the modified CEC-2010 test functions and the modified normal CEC-2010 test functions are equal but in the CEC-2013 LSGO benchmark functions, functions have uniform subcomponent sizes to create automatically different contribution for various subcomponents. In this study, the maximum number of evaluations was set to 3×10^6, the population size was set to 50, and all algorithms were evaluated for 25 independent runs and the results were recorded. Also, the dimension of all of the LSGO benchmark functions is set to 1000.

5.2 Experiment series 1: comparison among Bilevel framework and MOFBVE frameworks with higher number of levels, *k* (3, 4, and 5)

As mentioned earlier, an MOFBVE can be designed with the number of levels more than two levels (k-MOFBVE). In this section, we present the experimental results for three MOFBVE frameworks with 3, 4, and 5 levels and Bilevel framework and we have compared Bilevel against three MOFBVE frameworks. Also, an ideal k-MOFBVE with 3, 4, and 5 levels (k-MOFBVE-I) and Bilevel-I framework are designed which use the imbalance knowledge of a given problem to construct levels. A two-sided Wilcoxon statistical test with a confidence interval of 95 % is performed among the compared frameworks and the Bilevel framework. Symbols '†', '‡', and '≈' denote the compared frameworks are worse than, better than, or similar to Bilevel framework, respectively. *"w/t/l"* in the last row in tables means that the compared frameworks wins in w functions, ties in t functions, and loses in l functions, compared with Bilevel framework.

5.2.1 Experiment series 1. Part A: results on the modified CEC-2010 test functions

From Table 1 on the modified CEC-2010 test functions, 3-MOFBVE, 4-MOFBVE, and 5-MOFBVE achieve better results than Bilevel framework on 5 (f_{10},f_{13},f_{15}–f_{16} and f_{18}), 4 (f_{10}–f_{11}, and f_{15}–f_{16}), and 5 (f_{12}, and f_{15}–f_{18}) functions, respectively. Although Bilevel framework has better results than 3-MOFBVE, 4-MOFBVE, and 5-MOFBVE on 3 (f_9,f_{12}, and f_{14}), 3 (f_9,f_{14}, and f_{18}), 3 (f_9,f_{11}, and f_{14}) functions, respectively. The 3-MOFBVE, 4-MOFBVE, and 5-MOFBVE perform similar to the Bilevel framework on 2, 3, and 2 other functions, respectively. To gain a better understanding of the behavior of the Bilevel framework and k-MOFBVE algorithms, we plot the convergence graph on

Table 9 Results of Bilevel and Random Bilevel on the CEC-2013 LSGO benchmark test functions

Function		Bilevel-I	Bilevel	Random Bilevel
f_4	Mean	5.92e+09‡	9.09e+09$^{\approx}$	7.68e+09
	Std	2.14e+09	2.60e+09	2.56e+09
f_5	Mean	3.15e+06$^{\approx}$	2.69e+06‡	3.02e+06
	Std	4.87e+05	5.30e+05	5.90e+05
f_6	Mean	9.16e+04$^{\approx}$	8.56e+04$^{\approx}$	8.56e+04
	Std	1.63e+04	2.41e+04	3.02e+04
f_7	Mean	4.30e+06‡	5.86e+06‡	8.98e+06
	Std	1.08e+06	2.18e+06	1.28e+07
f_8	Mean	1.94e+13$^{\approx}$	2.31e+13$^{\approx}$	1.90e+13
	Std	7.84e+12	1.13e+13	8.27e+12
f_9	Mean	2.69e+08$^{\approx}$	2.81e+08$^{\approx}$	2.78e+08
	Std	2.70e+07	3.09e+07	3.61e+07
f_{10}	Mean	4.26e+04‡	3.34e+04‡	5.37e+04
	Std	1.51e+04	2.17e+04	5.89e+04
f_{11}	Mean	3.11e+08‡	7.64e+08†	5.58e+08
	Std	9.39e+07	1.04e+09	3.58e+08
f_{13}	Mean	5.15e+08‡	5.27e+08$^{\approx}$	5.30e+08
	Std	1.86e+08	1.48e+08	2.22e+08
f_{14}	Mean	9.67e+08†	6.90e+08†	3.63e+08
	Std	1.51e+09	1.02e+09	3.11e+08
	w/t/l	5/4/1	3/5/2	–

Symbols '†', '‡', and '≈' denote Bilevel is worse than, better than, or similar to Random Bilevel, respectively

four selected problems in Fig. 3. It can be seen from Table 2 that 3-MOFBVE-I, 4-MOFBVE-I, and 5-MOFBVE-I perform better than Bilevel-I framework on 4 (f_{11}–f_{13} and f_{16}), 5 (f_{11}–f_{13}, and f_{16}–f_{17}), and 5 (f_{11}–f_{13}, and f_{16}–f_{17}) functions, respectively. The Bilevel-I framework has the similar performance with 3-MOFBVE-I on 6 (f_9–f_{10}, f_{14}–f_{15} and f_{17}–f_{18}) functions and 4-MOFBVE-I and 5-MOFBVE-I on 5 (f_9–f_{10}, f_{14}–f_{15}, and f_{18}) functions. An important observation from Tables 1 and 2, is that the Bilevel framework performs well on the modified CEC-2010 test functions but the performance of MOFBVE is greatly enhanced when the number of the levels increases. One possible reason for that behavior is that with increase the number of levels, MOFBVE frameworks take the advantage of more important variables to explore promising regions. In addition, we observe that 3-MOFBVE-I, 4-MOFBVE-I, and 5-MOFBVE-I perform better than, or similar to the Bilevel-I framework.

5.2.2 *Experiment series 1. Part B: results on the modified normal CEC-2010 test functions*

On the modified normal CEC-2010 test functions, Table 3 indicates that 4-MOFBVE and 5-MOFBVE outperform Bilevel framework on 5 (f_9, f_{11}, f_{13}, and f_{15}–f_{16}) and 6 (f_{11}- f_{13}, and f_{15}–f_{17}) functions, respectively. 4-MOFBVE cannot perform better than Bilevel framework on only one function, f_{17}. The Bilevel framework has the similar performance with 4-MOFBVE and 5-MOFBVE on 3 and 4 other functions, respectively. 3-MOFBVE and Bilevel framework have the same results on all functions. To gain a better understanding of the behavior of the Bilevel framework and k-MOFBVE algorithms, we plot the convergence graph on four selected problems in Fig. 4. It can be seen from Table 4 that 3-MOFBVE-I and 4-MOFBVE-I can obtain the better results than Bilevel framework on 4 functions (f_{11}–f_{12}, f_{15}, and f_{17}). 5-MOFBVE-I achieves better results than Bilevel-I framework on 4 functions (f_{12} and f_{15}–f_{17}). 3-MOFBVE-I and 4-MOFBVE-I have the similar results in comparison with Bilevel-I framework on 6 functions (f_9–f_{10}, f_{13}–f_{14}, f_{16} and f_{18}). 5-MOFBVE-I performs similar to the Bilevel-I framework on 6 functions (f_9–f_{11}, f_{13}–f_{14}, and f_{18}). As it can be seen from Tables 3 and 4, the versions of k-MOFBVE frameworks with $k > 2$ have better than, or similar results to the Bilevel framework on the most functions in the modified normal CEC-2010 test functions. Once again, we can see that with increase the number of levels the k-MOFBVE has a more capability to improve its performance due to the use of more important variables as mentioned before.

5.2.3 *Experiment series 1. Part C: results on the CEC-2013 LSGO benchmark functions*

From Table 5, it can be seen that on the CEC-2013 LSGO benchmark functions, 3-MOFBVE, 4-MOFBVE, and 5-MOFBVE can obtain better results than Bilevel framework on 2 (f_8 and f_{10}), 1 (f_{10}), and 3 (f_4 and f_9–f_{10}) functions, respectively. 3-MOFBVE, 4-MOFBVE, and 5-MOFBVE have the same performance in comparison with Bilevel framework on 8, 9, and 7 other functions, respectively. It seems that Bilevel framework has the same performance with MOFBVE with k (3, 4, and 5) on the CEC-2013 LSGO benchmark functions. One possible reason for that behavior is that on some functions Morris method lose its efficiency due to new transformations and having subcomponents with non-uniform subcomponent sizes in these benchmark functions. Table 6 indicates that 3-MOFBVE-I, 4-MOFBVE-I, and 5-MOFBVE-I outperforms Bilevel-I framework on 3 functions (f_5 and f_9–f_{10}), 4 (f_7 and f_9–f_{11}), and 4 (f_7 and f_9–f_{11}) functions, respectively. 3-MOFBVE-I, 4-MOFBVE-I, and 5-MOFBVE-I perform similar to the Bilevel-I framework on 7, 6, and 6 other functions, respectively. As it can be seen from 6, when the levels of important variables is correctly constructed, MOFBVE frameworks with k (3, 4, and 5) have the capability to improve the performance of Bilevel framework. To gain a better understanding of the behavior of the Bilevel framework and k-MOFBVE

Table 10 Results of 4-MOFBVE and CC algorithms on the modified CEC-2010 test functions

Function		DECC-DG	CBCC1-DG	CBCC2-DG	4-MOFBVE
f_9	Mean	3.45e+11‡	2.75e+11‡	2.56e+11‡	1.44e+11
	Std	7.2e+10	4.42e+10	6.29e+10	2.46e+10
f_{10}	Mean	2.80e+07‡	2.78e+07‡	2.80e+07‡	1.03e+07
	Std	2.18e+06	2.22e+06	1.88e+06	1.92e+06
f_{11}	Mean	1.07e+01†	1.41e+01†	1.02e+01†	1.78e+04
	Std	7.67e−01	1.74e+01	9.1e−01	1.45e+04
f_{12}	Mean	5.12e+07‡	5.04e+07‡	4.63e+07‡	1.44e+06
	Std	3.19e+07	2.75e+07	2.13e+07	1.19e+06
f_{13}	Mean	1.06e+07$^{\approx}$	1.02e+07$^{\approx}$	1.08e+07$^{\approx}$	9.42e+06
	Std	3.14e+06	4.18e+06	5.03e+06	7.04e+06
f_{14}	Mean	6.43e+12‡	6.35e+12‡	6.13e+12‡	1.68e+12
	Std	1.41e+12	1.39e+12	1.42e+12	3.60e+11
f_{15}	Mean	1.98e+08‡	1.96e+08‡	2.01e+08‡	9.04e+07
	Std	8.05e+06	1.16e+07	8.82e+06	1.08e+07
f_{16}	Mean	2.15e−02†	2.11e−02†	1.83e−02†	5.77e+05
	Std	2.22e−02	2.6e−02	1.83e−02	4.16e+05
f_{17}	Mean	3.80e+09‡	4.20e+09‡	3.94e+09‡	3.52e+07
	Std	1.83e+09	2.00e+09	1.57e+09	5.71e+07
f_{18}	Mean	4.32e+08‡	5.36e+08‡	1.51e+08$^{\approx}$	5.57e+08
	Std	2.49e+08	3.85e+08	9.85e+07	1.97e+09
	w/t/l	7/1/2	7/1/2	6/1/3	–

Symbols '†', '‡', and '≈' denote 4-MOFBVE are worse than, better than, or similar to the compared algorithms, respectively

algorithms, we plot the convergence graph on four selected problems in Fig. 5.

5.2.4 Summary and discussion

The comparison results of the three MOFBVE frameworks with 3, 4, and 5 levels and Bilevel framework on the three benchmark suites can be summarized and explained as follows. From the study of Tables 1, 2, 3, 4, 5, and 6, it is clear that the versions of k-MOFBVE with $k > 2$ gained much better results than Bilevel framework. The above results demonstrate that k-MOFBVE can enhance the exploration ability to get better solutions. It can be observed that considering more important variables in three MOFBVE frameworks with 3, 4, and 5 levels are effective for optimizing imbalanced LSGO problems. As it can be seen from Tables 2, 3, 4, and 6, in comparison with Bilevel, the obtained results of 4-MOFBVE and 5-MOFBVE algorithms are better than 3-MOFBVE algorithm although Table 4 shows the algorithms have the same performance but by carefully looking at the results indicates that 4-MOFBVE and 5-MOFBVE algorithms achieve the mean values less than 3-MOFBVE algorithm on 6 and 4 functions, respectively. In addition, from Tables 2, 4, and 6 the performance of 4-MOFBVE and 5-MOFBVE is similar on all benchmark suites. Also, it is remarkable that k-MOFBVE can model better most real-world problems since they involve variables with a hierarchical levels of variant influences. Note that the main idea of MOFBVE framework is finding the variables with a significant effects therefore integrating the important variables at the beginning of LSGO algorithm can improve its performance. In order to show the importance of identifying variables with more effects on the fitness values, we compare Bilevel framework against random Bilevel framework. In the random Bilevel framework, 50 variables are randomly selected and replaced with the level of the important variables in Bilevel framework while other steps of random Bilevel framework is the same with Bilevel framework. The result derived from the random Bilevel framework and Bilevel are summarized in Tables 7, 8, and 9. From Tables 7 and 8, it can be seen that on the modified normal CEC-2010 test functions and the modified CEC-2010 test functions, Bilevel framework can obtain better results than the random Bilevel framework on 5 out of 10 functions. From Table 9, Bilevel and Bilevel-I frameworks can perform better than the random Bilevel framework on 3 and 5 out of 10 functions, respectively; although it achieves worse results than the random Bilevel framework on 2 and 1 out of 10 functions, respectively. As we mentioned earlier, Morris method has the poor performance on some functions due to the specific properties

Table 11 Results of 4-MOFBVE and CC algorithms on the modified CEC-2010 test functions

Function		DECC-I	CBCC1-I	CBCC2-I	4-MOFBVE-I
f_9	Mean	1.89e+11‡	1.88e+11‡	1.75e+11‡	1.11e+11
	Std	3.70e+10	4.35e+10	3.14e+10	2.43e+10
f_{10}	Mean	2.39e+07‡	2.31e+07‡	2.39e+07‡	1.06e+07
	Std	1.64e+06	2.51e+06	2.11e+06	1.50e+06
f_{11}	Mean	1.06e+01†	1.04e+01†	1.03e+01†	1.24e+04
	Std	1.04e+00	9.68e−01	8.10e−01	1.10e+04
f_{12}	Mean	2.91e+06‡	2.95e+06‡	3.76e+06‡	6.83e+05
	Std	1.04e+06	9.48e+05	3.43e+06	2.85e+05
f_{13}	Mean	1.73e+06†	1.44e+06†	1.09e+06†	1.06e+07
	Std	4.0e+05	5.42e+05	3.37e+05	3.62e+06
f_{14}	Mean	5.44e+12‡	6.96e+12‡	5.99e+12‡	1.59e+12
	Std	1.10e+12	2.92e+12	2.74e+12	3.55e+11
f_{15}	Mean	1.71e+08‡	1.72e+08‡	1.71e+08‡	9.72e+07
	Std	1.21e+07	1.36e+07	1.34e+07	1.06e+07
f_{16}	Mean	8.61e−09†	7.75e−09†	1.06e−08†	4.70e+05
	Std	1.92e−09	1.56e−09	3.35e−09	2.11e+05
f_{17}	Mean	4.70e+08‡	7.73e+08‡	5.92e+08‡	1.61e+07
	Std	8.80e+07	3.76e+08	3.26e+08	7.06e+06
f_{18}	Mean	3.29e+07†	3.28e+07†	2.24e+07†	1.39e+08
	Std	1.57e+07	1.52e+07	1.01e+07	1.05e+08
	w/t/l	6/0/4	6/0/4	6/0/4	–

Symbols '†', '‡', and '≈' denote 4-MOFBVE are worse than, better than, or similar to the compared algorithms, respectively

of the CEC-2013 LSGO benchmark functions. Although, we can see that in the Bilevel-I framework when the important variables is correctly are identified, Bilevel-I can obtain better than or comparable results to the random Bilevel framework on the most of the test functions. In overall, it is demonstrated that identifying important variables has a remarkable impact on enhancing performance of Bilevel framework.

5.3 Experiment series 2: a comparison of 4-MOFBVE framework with the CC algorithms

In this section, we selected 4-MOFBVE among three MOFBVE with 3, 4, and 5 levels to compare it against several CC algorithms. On the modified CEC-2010 test functions and the modified normal CEC-2010 test functions, 4-MOFBVE framework is compared with CC algorithms with DG decomposition method. Also, the 4-MOFBVE ideal framework (4-MOFBVE-I) is compared with the CC algorithms with the ideal decomposition method which uses the priori knowledge about the benchmark functions to construct subcomponents. 4-MOFBVE-I uses the imbalance knowledge of a given problem to recognize the important variables. The CC algorithms with the ideal grouping utilize the nonseparable knowledge of a given problem such that they can construct all subcomponents. Due to the specific properties of the CEC-2013 LSGO benchmark functions, as mentioned above, DG decomposition method performs poorly on these functions; thus we compare these functions with the CC algorithms with the ideal decomposition method. Ideal grouping method constructs subcomponents manually using the knowledge of benchmark functions. CC algorithms use the round-robin strategy (DECC) and CBCC (CBCC1 and CBCC2) (Omidvar et al. 2011, 2014a) strategies which assign more budget for the subcomponent with the most effect. A two-sided Wilcoxon statistical test with a confidence interval of 95 % is conducted among CC algorithms and 4-MOFBVE framework. Symbols '†', '‡', and '≈' denote 4-MOFBVE are worse than, better than, or similar to the compared algorithms, respectively. *"w/t/l"* in the last row in tables means that Bilevel framework wins in *w* functions, ties in *t* functions, and loses in *l* functions, compared with CC algorithms.

5.3.1 *Experiment series 2. Part A: results on the modified CEC-2010 test functions*

Table 10 presents the results of 4-MOFBVE framework to compare with the CC algorithms with along DG decomposition method with round-robin strategy (DECC-DG), CBCC1 (CBCC1-DG), and CBCC2 (CBCC2-DG) budget division methods on the modified CEC-2010 test functions. It can be

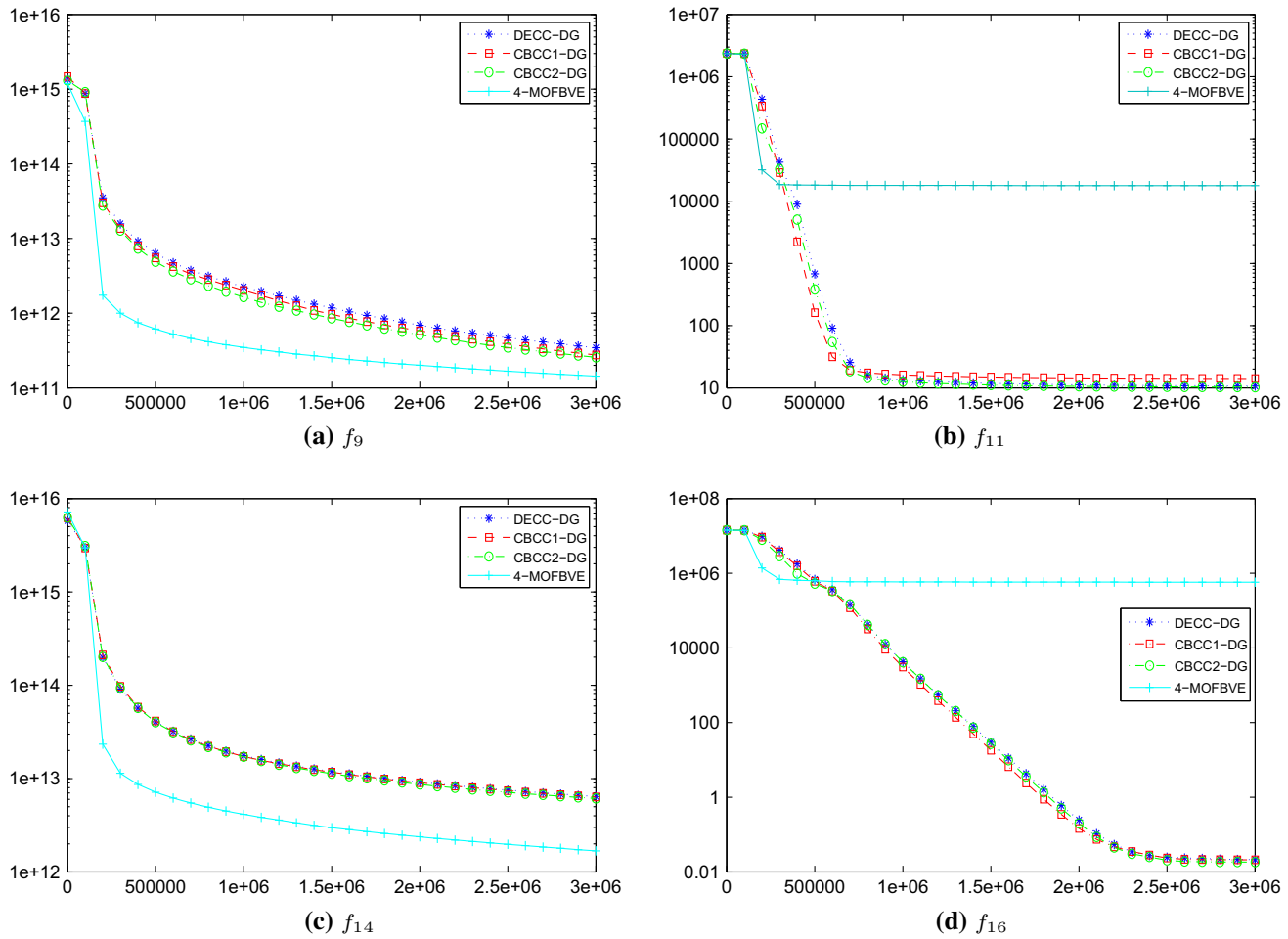


Fig. 6 Convergence plots of f_9, f_{11}, f_{14}, and f_{16} of the modified CEC-2010 test functions. The results were averaged over 25 runs and 4-MOFBVE framework is compared with the CC algorithms using DG decomposition method. The vertical axis is the function value and the horizontal axis is the number of function evaluations

seen from Table 10 that 4-MOFBVE framework outperforms DECC-DG and CBCC1-DG on 7 functions (f_9–f_{10}, f_{12}, f_{14}–f_{15}, and f_{17}–f_{18}); but DECC-DG and CBCC1-DG can obtain better results than 4-MOFBVE framework on 2 functions (f_{11} and f_{16}). 4-MOFBVE framework performs better than CBCC2-DG on 6 functions (f_9–f_{10}, f_{12}, f_{14}–f_{15}, and f_{17}) while it cannot achieve better results than CBCC2-DG on 3 functions (f_{11}, f_{16}, and f_{18}). The 4-MOFBVE framework has the same results in comparison with the compared methods on only the function f_{13}. In addition, the results of the CC algorithms with the ideal grouping and 4-MOFBVE-I are summarized in Table 11. It is obvious from Table 11 on the modified CEC-2010 test functions that 4-MOFBVE-I performs better than DECC-I, CBCC1-I, and CBCC2-I on 6 functions (f_9–f_{10}, f_{12}, f_{14}–f_{15}, and f_{17}) while it cannot achieve better results than the compared cc algorithms on 4 functions (f_{11}, f_{13}, f_{16}, and f_{18}).

It is remarkable that the 4-MOFBVE framework has the worse results compared to the CC algorithms especially on the most functions (f_{11} and f_{16}) where Ackley's functions are used to form their nonseparable subcomponents on these test benchmark function. To gain a better understanding of the behavior of the 4-MOFBVE framework and CC algorithms, we plot the convergence graph on four selected problems (two functions per class of functions) with the DG decomposition method in Fig. 6. The 4-MOFBVE-I framework uses only three levels of important subcomponents for more optimization for several iterations so the 4-MOFBVE-I framework is approximately an non-decomposition-based method, i.e., it tackles LSGO problems approximately as a whole. The CC algorithms take the advantage of the divide-and-conquer strategy to divide LSGO problem into several low-dimensional subcomponents. The reason that the performance of the 4-MOFBVE framework is either worse than or the same as compared CC methods on some functions might be due to the type of its strategy (non-decomposition) to handle LSGO problems. Furthermore, the comparison between the ideal CC algorithms and the 4-MOFBVE-I framework

Table 12 Results of 4-MOFBVE and CC algorithms on the modified normal CEC-2010 test functions

Function		DECC-DG	CBCC1-DG	CBCC2-DG	4-MOFBVE
f_9	Mean	1.1e+11‡	1.22e+11‡	1.31e+11‡	3.31e+10
	Std	6.44e+10	5.03e+10	5.18e+10	8.20e+09
f_{10}	Mean	3.04e+05‡	3.03e+05‡	3.00e+05‡	1.96e+05
	Std	1.54e+04	1.78e+04	1.93e+04	2.37e+04
f_{11}	Mean	1.05e+01†	1.03e+01†	1.10e+01†	1.36e+03
	Std	6.51e−01	6.92e−01	9.07e−01	5.88e+02
f_{12}	Mean	1.60e+07‡	1.74e+07‡	1.44e+07‡	1.20e+06
	Std	5.01e+06	4.23e+06	5.86e+06	6.69e+05
f_{13}	Mean	1.61e+07$^{\approx}$	1.61e+07$^{\approx}$	1.42e+07$^{\approx}$	1.25e+07
	Std	8.36e+06	9.02e+06	7.19e+06	8.05e+06
f_{14}	Mean	2.27e+13‡	2.45e+13‡	2.14e+13‡	3.55e+12
	Std	8.58e+12	6.46e+12	7.76e+12	7.82e+11
f_{15}	Mean	5.90e+07‡	5.97e+07‡	5.91e+07‡	2.86e+07
	Std	4.22e+06	3.88e+06	3.67e+06	4.43e+06
f_{16}	Mean	1.13e+00†	6.04e−01†	1.64e+01†	1.26e+05
	Std	4.19e−01	1.30e−01	7.79e+01	8.87e+04
f_{17}	Mean	1.54e+10‡	2.34e+10‡	1.66e+10‡	8.14e+07
	Std	7.32e+09	7.40e+09	6.89e+09	9.97e+07
f_{18}	Mean	7.84e+10‡	9.64e+10‡	1.06e+11‡	1.89e+10
	Std	1.05e+11	1.70e+11	1.93e+11	9.92e+09
	w/t/l	7/1/2	7/1/2	7/1/2	–

Symbols '†', '‡', and '≈' denote 4-MOFBVE are worse than, better than, or similar to the compared algorithms, respectively

is not totally fair comparison because the 4-MOFBVE-I framework takes only the imbalance knowledge about three important subcomponents while the ideal CC algorithms takes the nonseparable knowledge of all variables.

5.3.2 Experiment series 2. Part B: results on the modified normal CEC-2010 test functions

Table 12 shows the results of 4-MOFBVE framework to compare with CC algorithms with along DG decomposition method with the round-robin (DECC-DG), CBCC1 (CBCC1-DG), and CBCC2 (CBCC2-DG) budget division methods on the modified normal CEC-2010 test functions. It is obvious from Table 12 that 4-MOFBVE framework outperforms DECC-DG, CBCC1-DG, and CBCC2-DG on 7 (f_9–f_{10},f_{12},f_{14}–f_{15}, and f_{17}–f_{18}), 7 (f_9–f_{10},f_{12},f_{14}–f_{15}, and f_{17}–f_{18}), and 7 (f_9–f_{10},f_{12},f_{14}–f_{15}, and f_{17}–f_{18}) functions, respectively. However, 4-MOFBVE framework cannot outperform DECC-DG, CBCC1-DG, and CBCC2-DG on 2 (f_{11} and f_{16}). The 4-MOFBVE framework achieves the same results in comparison with DECC-DG and CBCC1-DG on the function f_{13} and CBCC1-DG on the function f_{13}. Also, the results of the CC algorithms with the ideal grouping and 4-MOFBVE-I are summarized in Table 13. It is obvious from Table 13 that 4-MOFBVE-I performs better than DECC-I on 6 functions (f_9–f_{10},f_{12},f_{14}–f_{15}, and f_{17}) while it cannot achieve better results than DECC-I on 4 functions (f_{11},f_{13},f_{16}, and f_{18}). 4-MOFBVE-I outperforms CBCC1-I and CBCC2-I on 5 functions (f_9–f_{10},f_{14}–f_{15}, and f_{17}); but CBCC1-I and CBCC2-I obtain better results than 4-MOFBVE-I on 5 functions (f_{11}–f_{13},f_{16}, and f_{18}). By comparing the performance of the 4-MOFBVE-I framework and the compared CC algorithms with the ideal decomposition method, it can be seen that both algorithms perform similarly on the modified normal CEC-2010 test functions. As mentioned above, one possible reason for the poor performance of the 4-MOFBVE-I framework on some functions might be due to the type of its strategy (non-decomposition) to handle LSGO problems. Furthermore, the comparison between the ideal CC algorithms and the 4-MOFBVE-I framework is not totally fair comparison and the 4-MOFBVE-I uses the low level of the prior knowledge than the ideal CC algorithms as mentioned above. It is notable that, like the modified CEC-2010 test functions, 4-MOFBVE framework also has same behavior on f_{11} and f_{16}, Ackley's functions are used to form their nonseparable subcomponents, and 4-MOFBVE framework was trapped in a local optimum. To gain a better understanding of the behavior of the 4-MOFBVE framework and CC algorithms, we plot the convergence graph on four selected problems (two functions

Table 13 Results of 4-MOFBVE and CC algorithms on the modified normal CEC-2010 test functions

Function		DECC-I	CBCC1-I	CBCC2-I	4-MOFBVE-I
f_9	Mean	8.15e+10‡	3.92e+10‡	3.96e+10‡	2.50e+10
	Std	2.46e+10	1.27e+10	1.24e+10	6.91e+09
f_{10}	Mean	3.12e+05‡	2.83e+05‡	2.83e+05‡	2.12e+05
	Std	1.94e+04	1.79e+04	1.55e+04	2.60e+04
f_{11}	Mean	1.03e+01†	1.03e+01†	1.06e+01†	9.71e+02
	Std	1.02e+00	1.03e+00	1.08e+00	1.28e+02
f_{12}	Mean	1.83e+07‡	3.81e+05†	3.84e+05†	7.19e+05
	Std	6.04e+06	9.10e+04	9.03e+04	3.14e+05
f_{13}	Mean	8.94e+06†	2.14e+06†	5.62e+05†	2.10e+07
	Std	4.90e+06	1.08e+06	4.85e+05	7.91e+06
f_{14}	Mean	9.43e+13‡	2.62e+13‡	2.18e+13‡	3.49e+12
	Std	2.92e+13	9.01e+12	4.93e+12	9.99e+11
f_{15}	Mean	6.36e+07‡	5.61e+07‡	5.55e+07‡	2.73e+07
	Std	3.34e+06	3.76e+06	4.50e+06	3.86e+06
f_{16}	Mean	1.48e−08†	1.12e−08†	1.30e−08†	1.12e+05
	Std	2.52e−09	1.07e−09	5.26e−09	8.30e+04
f_{17}	Mean	1.45e+11‡	6.50e+09‡	3.97e+09‡	1.66e+07
	Std	3.36e+10	1.94e+09	1.96e+09	1.22e+07
f_{18}	Mean	1.45e+10†	9.87e+09†	9.23e+09†	2.54e+10
	Std	5.37e+09	4.17e+09	4.88e+09	9.29e+09
	w/t/l	6/0/4	5/0/5	5/0/5	–

Symbols '‡', '†', and '≈' denote 4-MOFBVE are worse than, better than, or similar to the compared algorithms, respectively

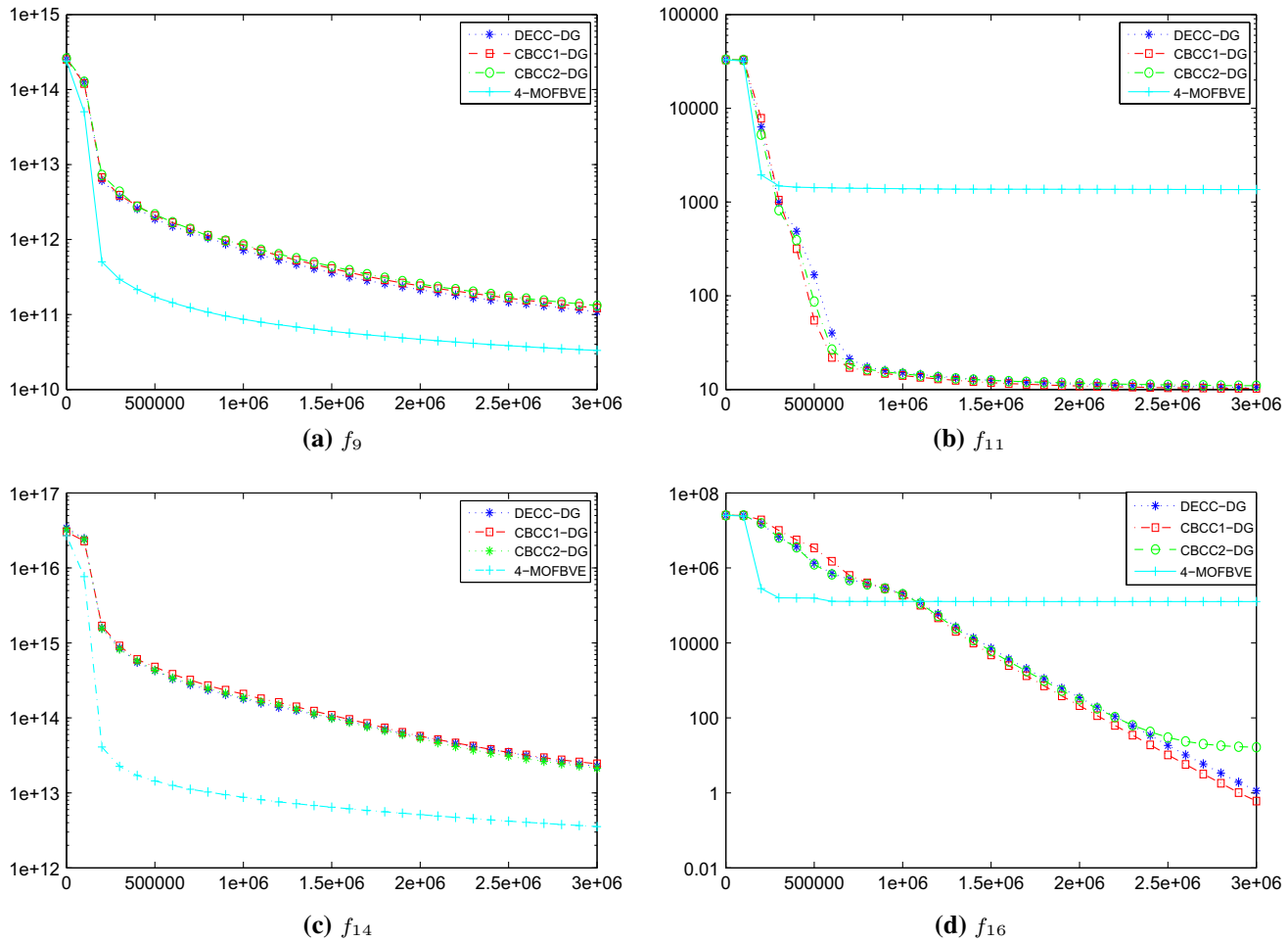
per class of functions) with the DG decomposition method in Fig. 7.

5.3.3 *Experiment series 2. Part C: results on the CEC-2013 LSGO benchmark functions*

The results of 4-MOFBVE framework to compare with CC algorithms with along ideal decomposition method with round-robin (CC-ideal), CBCC1 (CBCC1-ideal), and CBCC2 (CBCC2-ideal) budget division methods on the CEC-2013 LSGO benchmark functions are summarized in Table 14. It can be seen from Table 14 that 4-MOFBVE framework outperforms CC-ideal, CBCC1-ideal, and CBCC2-ideal on 8 functions (f_4–f_5, f_7–f_9, and f_{11}–f_{13}). However, the compared CC methods with ideal decomposition method can obtain better results than 4-MOFBVE framework on 2 functions (f_6 and f_{10}). As we mentioned earlier the 4-MOFBVE framework method becomes ineffective on most functions having Ackley nonseparable subcomponents such as f_6 and f_{10} in these test functions. Although, CC algorithms with along ideal decomposition method used the prior knowledge of the benchmark functions to decompose functions, 4-MOFBVE framework still outperforms them on most of functions. CC methods lose its effectiveness on the CEC-2013 LSGO benchmark functions due to the specific properties of the CEC-2013 LSGO benchmark functions, as mentioned above. Also, the convergence plots for four selected functions on the CEC-2013 LSGO benchmark functions (two functions from two classes of functions: functions with seven imbalanced subcomponents and functions with 20 imbalanced subcomponents) are shown in Fig. 8.

5.3.4 *Summary and discussion*

The comparison results of 4-MOFBVE framework and CC algorithms on the three benchmark suites can be summarized and explained as follows. From the study of Tables 7, 9, and 11, it can be seen that 4-MOFBVE framework is significantly better than CC algorithms with the various budget division methods although the performance of the 4-MOFBVE framework decreases on the most functions having Ackley nonseparable subcomponents. In 4-MOFBVE framework, three levels of important variables are optimized before an LSGO algorithm to enhance the ability of an LSGO algorithm to handle the imbalanced LSGO problems. Besides the above statistical tests, in order to deeply analyze the performance differences of 4-MOFBVE framework with regard to DECC, CCBC1, and CCBC2 algorithms, we also conduct



(a) f_9 **(b)** f_{11} **(c)** f_{14} **(d)** f_{16}

Fig. 7 Convergence plots of f_9, f_{11}, f_{14}, and f_{16} of the modified normal CEC-2010 test functions. The results were averaged over 25 runs and 4-MOFBVE framework is compared with the CC algorithms using DG decomposition method. The vertical axis is the function value and the horizontal axis is the number of function evaluations

a pairwise signed-rank Wilcoxon test (Wilcoxon 1945). The pairwise signed-rank Wilcoxon test are used to compare the performance of two algorithms on a common set of problems (Derrac et al. 2011). Here, on all benchmark functions, the total number of functions is 30. Tables 15, 16, and 17 show the ranks and p-values of Wilcoxon test between 4-MOFBVE framework and CC algorithms. A plus sign '+' indicates that 4-MOFBVE framework statistically performs better than other algorithms at significance level 5 %. We observe that the rank values corresponding to 4-MOFBVE framework are always greater than other algorithms.

5.4 Experiment series 3: comparison of MOFBVE algorithms with SaNSDE algorithm

In this section, we compare SaNSDE against 3-MOFBVE, 4-MOFBVE, 5-MOFBVE, and the Bilevel framework. SaNSDE is applied as the subcomponent optimization algorithm in the MOFBVE algorithms. This comparison aim is investigating whether all MOFBVE frameworks perform better than the internal parent optimizer. A two-sided Wilcoxon statistical test with a confidence interval of 95 % is performed among the compared frameworks and SaNSDE. Symbols '†', '‡', and '≈' denote the compared frameworks are worse than, better than, or similar to SaNSDE, respectively. *"w/t/l"* in the last row in tables means that the compared frameworks wins in w functions, ties in t functions, and loses in l functions, compared with SaNSDE. The result derived from the compared frameworks and SaNSDE are summarized in Tables 18, 19, and 20. It can be seen from Table 18 that on the modified CEC-2010 test functions, 3-MOFBVE, 4-MOFBVE, 5-MOFBVE, and the Bilevel framework can obtain better results than SaNSDE on 6 (f_9,f_{11}−f_{12},f_{14} and f_{16}−f_{17}), 5 (f_{10}−f_{11} and f_{15}−f_{17}), 7 (f_{10}−f_{13} and f_{15}−f_{17}), and 7 (f_{10}−f_{12} and f_{15}−f_{18}) functions, respectively. 4-MOFBVE achieves worse result

Table 14 Results of 4-MOFBVE and CC algorithms on the CEC-2013 LSGO benchmark test functions

Function		CC-ideal	CBCC1-ideal	CBCC2-ideal	4-MOFBVE
f_4	Mean	4.97e+10‡	2.60e+10‡	2.76e+10‡	9.13e+09
	Std	1.97e+10	1.31e+10	1.43e+10	3.079e+09
f_5	Mean	4.96e+06‡	4.42e+06‡	4.40e+06‡	2.80e+06
	Std	3.63e+05	4.56e+05	4.24e+05	3.61e+05
f_6	Mean	1.2e+04†	4.07e+03†	1.44e+04†	9.11e+04
	Std	2.51e+04	1.41e+04	3.13e+04	2.15e+04
f_7	Mean	6.33e+07‡	4.58e+07‡	5.46e+07‡	9.35e+06
	Std	2.36e+07	2.37e+07	2.49e+07	1.26e+07
f_8	Mean	4.86e+15‡	2.87e+15‡	2.48e+15‡	2.54e+13
	Std	1.85e+15	1.15e+15	9.83e+14	1.53e+13
f_9	Mean	4.97e+08‡	4.30e+08‡	4.28e+08‡	2.62e+08
	Std	3.53e+07	2.28e+07	4.06e+07	2.62e+07
f_{10}	Mean	1.64e+01†	1.44e+01†	8.88e+00†	2.55e+03
	Std	1.96e+01	1.84e+01	1.05e+01	3.31e+02
f_{11}	Mean	3.27e+09‡	2.76e+09‡	2.25e+09‡	4.48e+08
	Std	5.11e+09	2.56e+09	1.97e+09	3.97e+08
f_{13}	Mean	8.96e+09‡	8.85e+09‡	8.46e+09‡	5.68e+08
	Std	2.3e+09	3.06e+09	2.99e+09	2.19e+08
f_{14}	Mean	8.77e+10‡	7.25e+10‡	7.35e+10‡	7.57e+08
	Std	2.53e+10	2.78e+10	3.16e+10	1.53e+09
	w/t/l	8/0/2	8/0/2	8/0/2	–

Symbols '†', '‡', and '≈' denote 4-MOFBVE are worse than, better than, or similar to the compared algorithms, respectively

than SaNSDE on only one function f_9. 3-MOFBVE, 4-MOFBVE, 5-MOFBVE, and the Bilevel framework have the similar performance in comparison with SaNSDE on 4, 5, 2, and 3 other functions. On the modified normal CEC-2010 test functions, Table 19 indicates that 3-MOFBVE, 4-MOFBVE, 5-MOFBVE, and the Bilevel framework outperform SaNSDE on 3 (f_{11}–f_{12} and f_{16}), 3 (f_{11}, f_{16}, and f_{18}), 4 (f_{11},f_{15}–f_{16} and f_{18}), and 6 (f_{11}–f_{13} and f_{16}–f_{18}) functions, respectively. 3-MOFBVE, 4-MOFBVE, and 5-MOFBVE achieve worse results than SaNSDE on 1 (f_9), 3 (f_9 and f_{13}–f_{14}), and 1 (f_{14}) functions, respectively. 3-MOFBVE, 4-MOFBVE, 5-MOFBVE, and the Bilevel framework similar results in comparison with SaNSDE on 7 (f_9–f_{10},f_{13}–f_{15}, and f_{17}–f_{18}), 6 (f_{10}, f_{12}–f_{15}, and f_{17}), 3 (f_{10},f_{12}, and f_{17}), and 3 (f_9–f_{10}, and f_{15}) functions, respectively.

It is obvious from Table 20 on the CEC-2013 LSGO benchmark functions that 3-MOFBVE, 4-MOFBVE, 5-MOFBVE, and the Bilevel framework perform better than SaNSDE on 1 (f_5), 2 (f_5 and f_{10}), 1 (f_{10}), and 2 (f_9–f_{10}) functions, respectively. 3-MOFBVE, 4-MOFBVE, 5-MOFBVE, and the Bilevel framework achieve the same results in comparison with SaNSDE on 9, 8, 9, and 8 other functions. As we mentioned earlier Morris method performed poorly on some functions in the CEC-2013 LSGO benchmark suit therefore SaNSDE is compared with the ideal k-MOFBVE with 3, 4, and 5 levels and Bilevel-I. From Table 21, we can observe that 3-MOFBVE-I, 4-MOFBVE-I, 5-MOFBVE-I, and the Bilevel-I framework can obtain better results than SaNSDE on 5 (f_4,f_7, and f_9–f_{11}), 5 (f_4,f_7, and f_9–f_{11}), 4 (f_7, and f_9–f_{11}), and 2 (f_4 and f_7) functions, respectively. 3-MOFBVE, 4-MOFBVE, 5-MOFBVE, and the Bilevel framework achieve the same results in comparison with SaNSDE on 5, 5, 4, and 8 other functions. From the study of Tables 18, 19, 20, and 21, we can conclude that the performance of k-MOFBVE is either significantly better than or comparable to SaNSDE as the parent algorithm. Important observations form results is that the performance of k-MOFBVE is significantly better than SaNSDE when the number of the levels in the k-MOFBVE increases. The main reason why k-MOFBVE performs better than SaNSDE is that it reduce the search space of an imbalanced LSGO problem to a small search space with the most important variables to find effective direction toward promising regions.

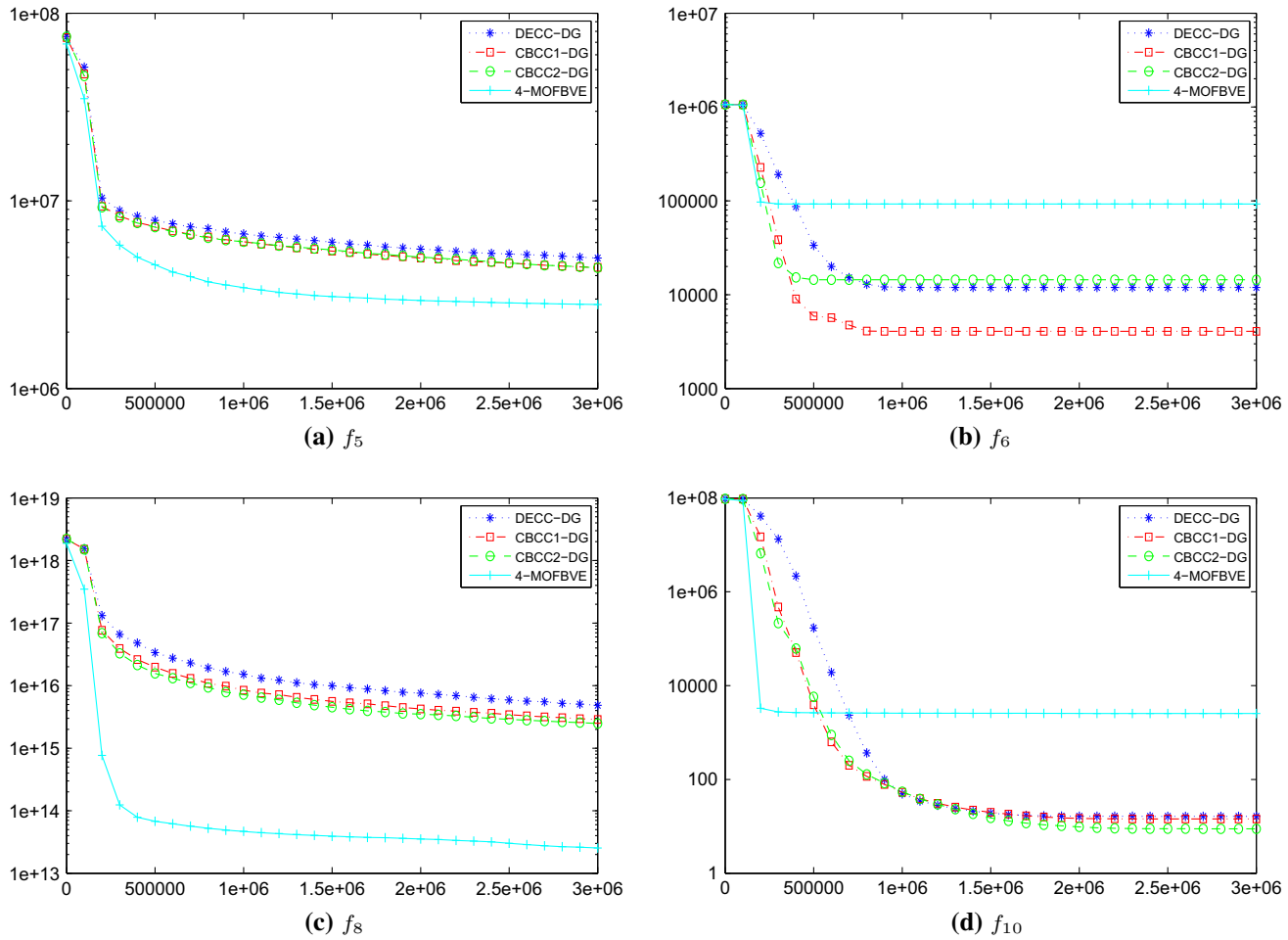


Fig. 8 Convergence plots of f_5, f_6, f_8, and f_{10} of the CEC-2013 LSGO benchmark functions. The results were averaged over 25 runs and 4-MOFBVE framework is compared with the CC algorithms using the ideal decomposition method. The vertical axis is the function value and the horizontal axis is the number of function evaluations

Table 15 Wilcoxon test between 4-MOFBVE and DECC

4-MOFBVE.Ranks	425
DECC-DG.Ranks	40
p value	8e−05
Results at the 5 % significance level	+

Table 16 Wilcoxon test between 4-MOFBVE and CCBC1

4-MOFBVE.Ranks	429
CCBC1-DG.Ranks	36
p value	0
Results at the 5 % significance level	+

Table 17 Wilcoxon test between 4-MOFBVE and CCBC2

4-MOFBVE.Ranks	424
CCBC2-DG.Ranks	41
p value	8e−05
Results at the 5 % significance level	+

6 Conclusions and future directions

In this paper, we proposed a new optimization framework to solve LSGO problems more efficiently. MOFBVE has a simple concept, a hierarchical model composed of the several levels with the specific search space of the optimization problem. In MOFBVE, the search space associated with an LSGO problem is transformed to a low-dimension search space including variables with the most influence on the fitness value to obtain fitter initial sub-solutions for the last level with the original search space. The main motivation of MOFBVE relies on the complexity reduction of the search space in the LSGO problems. The sensitivity analysis methods are used to detect the influence of the variables on the objective function. In MOFBVE, when the number of the levels increases, the performance of MOFBVE is enhanced sig-

Table 18 Results of SaNSDE, Bilevel, and MOFBVE with k (3, 4, and 5) on the modified CEC-2010 test functions

Function		Bilevel	3-MOFBVE	4-MOFBVE	5-MOFBVE	SaNSDE
f_9	Mean	1.08e+11‡	1.42e+11$^{\approx}$	1.44e+11†	1.36e+11$^{\approx}$	1.24e+11
	Std	1.80e+10	2.68e+10	2.46e+10	2.62e+10	3.12e+10
f_{10}	Mean	1.18e+07$^{\approx}$	1.04e+07‡	1.03e+07‡	1.11e+07‡	1.19e+07
	Std	1.30e+06	2.02e+06	1.92e+06	1.33e+06	1.07e+06
f_{11}	Mean	2.57e+04‡	2.66e+04‡	1.78e+04‡	2.68e+04‡	1.77e+05
	Std	8.83e+03	2.32e+04	1.45e+04	5.20e+04	8.90e+04
f_{12}	Mean	1.12e+06‡	1.36e+06$^{\approx}$	1.44e+06‡	5.26e+05‡	1.55e+06
	Std	5.88e+05	5.38e+05	1.19e+06	1.72e+05	5.89e+05
f_{13}	Mean	1.25e+07$^{\approx}$	1.11e+07$^{\approx}$	9.42e+06‡	1.20e+07$^{\approx}$	1.16e+07
	Std	7.69e+06	5.91e+06	7.04e+06	8.24e+06	4.25e+06
f_{14}	Mean	1.40e+12‡	1.64e+12$^{\approx}$	1.68e+12$^{\approx}$	1.63e+12$^{\approx}$	1.54e+12
	Std	2.23e+11	3.24e+11	3.60e+11	2.86e+11	2.70e+11
f_{15}	Mean	1.02e+08$^{\approx}$	9.67e+07‡	9.04e+07‡	9.55e+07‡	1.04e+08
	Std	1.12e+07	9.46e+06	1.08e+07	1.32e+07	1.07e+07
f_{16}	Mean	1.28e+06‡	9.54e+05‡	5.77e+05‡	6.55e+05‡	2.23e+06
	Std	2.51e+05	4.71e+05	4.16e+05	5.26e+05	9.23e+05
f_{17}	Mean	2.84e+07‡	3.68e+07‡	3.52e+07‡	2.01e+07‡	4.15e+07
	Std	1.60e+07	4.83e+07	5.71e+07	1.34e+07	3.96e+07
f_{18}	Mean	3.00e+08$^{\approx}$	2.36e+08$^{\approx}$	5.57e+08†	1.49e+08‡	2.23e+08
	Std	1.91e+08	1.87e+08	1.97e+09	6.36e+07	9.69e+07
	w/t/l	6/4/0	5/5/0	7/2/1	7/3/0	–

Symbols '†', '‡', and '≈' denote the compared algorithms are worse than, better than, or similar to SaNSDE, respectively

Table 19 Results of SaNSDE, Bilevel, and MOFBVE with k (3, 4, and 5) on the modified normal CEC-2010 test functions

Function		Bilevel	3-MOFBVE	4-MOFBVE	5-MOFBVE	SaNSDE
f_9	Mean	2.57e+10$^{\approx}$	3.18e+10†	3.31e+10†	2.85e+10$^{\approx}$	2.43e+10
	Std	6.68e+09	8.86e+09	8.20e+09	8.12e+09	4.89e+09
f_{10}	Mean	2.05e+05$^{\approx}$	2.06e+05$^{\approx}$	1.96e+05$^{\approx}$	1.97e+05$^{\approx}$	1.98e+05
	Std	2.68e+04	2.43e+04	2.37e+04	3.09e+04	2.68e+04
f_{11}	Mean	1.84e+03‡	1.54e+03‡	1.36e+03‡	1.06e+03‡	3.03e+03
	Std	5.67e+02	7.25e+02	5.88e+02	4.48e+02	5.56e+02
f_{12}	Mean	9.50e+05‡	1.30e+06$^{\approx}$	1.20e+06$^{\approx}$	6.89e+05‡	1.71e+06
	Std	2.91e+05	9.91e+05	6.69e+05	4.36e+05	2.05e+06
f_{13}	Mean	1.99e+07$^{\approx}$	1.45e+07$^{\approx}$	1.25e+07†	1.12e+07‡	1.92e+07
	Std	1.04e+07	1.10e+07	8.05e+06	9.12e+06	8.25e+06
f_{14}	Mean	3.36e+12$^{\approx}$	3.46e+12$^{\approx}$	3.55e+12†	3.56e+12†	3.11e+12
	Std	9.65e+11	7.97e+11	7.82e+11	8.13e+11	1.01e+12
f_{15}	Mean	3.25e+07$^{\approx}$	3.06e+07$^{\approx}$	2.86e+07‡	3.02e+07$^{\approx}$	3.15e+07
	Std	4.26e+06	3.98e+06	4.43e+06	2.78e+06	3.21e+06
f_{16}	Mean	2.28e+05‡	1.95e+05‡	1.26e+05‡	1.76e+05‡	1.56e+06
	Std	9.74e+04	1.94e+05	8.87e+04	3.30e+05	1.01e+06
f_{17}	Mean	4.07e+07$^{\approx}$	8.61e+07$^{\approx}$	8.14e+07$^{\approx}$	1.60e+07‡	5.96e+07
	Std	5.70e+07	1.26e+08	9.97e+07	1.60e+07	9.93e+07
f_{18}	Mean	3.24e+10$^{\approx}$	2.40e+10†	2.34e+10†	2.50e+10†	5.96e+07
	Std	2.49e+10	1.62e+10	9.92e+09	1.10e+10	2.71e+10
	w/t/l	3/7/0	2/6/2	3/3/4	5/3/2	–

Symbols '†', '‡', and '≈' denote the compared algorithms are worse than, better than, or similar to SaNSDE, respectively

Table 20 Results of SaNSDE, Bilevel, and MOFBVE with k (3, 4, and 5) on the CEC-2013 LSGO benchmark test functions

Function		Bilevel	3-MOFBVE	4-MOFBVE	5-MOFBVE	SaNSDE
f_4	Mean	9.09e+09$^{\approx}$	9.92e+09$^{\approx}$	9.13e+09$^{\approx}$	7.29e+09$^{\approx}$	8.23e+09
	Std	2.60e+09	5.01e+09	3.08e+09	3.63e+09	3.09e+09
f_5	Mean	2.69e+06‡	2.77e+06‡	2.80e+06$^{\approx}$	3.01e+06$^{\approx}$	3.01e+06
	Std	5.30e+05	4.19e+05	3.61e+05	5.43e+05	4.25e+05
f_6	Mean	8.56e+04$^{\approx}$	9.33e+04$^{\approx}$	9.25e+04$^{\approx}$	8.81e+04$^{\approx}$	9.31e+04
	Std	2.41e+04	2.91e+04	2.15e+04	2.59e+04	1.49e+04
f_7	Mean	5.85e+06$^{\approx}$	5.82e+06$^{\approx}$	9.35e+06$^{\approx}$	6.53e+06$^{\approx}$	6.12e+06
	Std	2.18e+06	2.21e+06	1.26e+07	2.80e+06	2.65e+06
f_8	Mean	2.31e+13$^{\approx}$	1.81e+13$^{\approx}$	2.54e+13$^{\approx}$	2.88e+13$^{\approx}$	2.23e+13
	Std	8.86e+12	9.05e+12	1.03e+13	1.15e+13	1.50e+13
f_9	Mean	2.81e+08$^{\approx}$	2.65e+08$^{\approx}$	2.62e+08$^{\approx}$	1.94e+08‡	2.72e+08
	Std	3.09e+07	3.16e+07	2.62e+07	3.30e+07	2.49e+07
f_{10}	Mean	3.34e+04$^{\approx}$	2.09e+04‡	2.55e+03‡	1.89e+03‡	4.42e+04
	Std	2.17e+04	2.22e+04	3.31e+02	1.15e+03	1.25e+04
f_{11}	Mean	7.64e+08$^{\approx}$	4.55e+08$^{\approx}$	4.48e+08$^{\approx}$	3.83e+09$^{\approx}$	3.90e+08
	Std	1.04e+09	3.66e+08	3.97e+08	1.34e+10	2.01e+08
f_{13}	Mean	5.27e+08$^{\approx}$	5.31e+08$^{\approx}$	5.68e+08$^{\approx}$	6.56e+08$^{\approx}$	5.19e+08
	Std	1.48e+08	1.95e+08	2.19e+08	2.80e+08	2.83e+08
f_{14}	Mean	6.90e+08‡	2.15e+09$^{\approx}$	7.57e+08$^{\approx}$	6.46e+08$^{\approx}$	4.34e+08
	Std	1.02e+09	8.52e+09	1.53e+09	7.15e+08	4.63e+08
	w/t/l	1/8/1	2/8/0	1/9/0	2/8/0	–

Symbols '†', '‡', and '≈' denote the compared algorithms are worse than, better than, or similar to SaNSDE, respectively

Table 21 Results of SaNSDE, Bilevel-I, and MOFBVE-I with k (3, 4, and 5) on the CEC-2013 LSGO benchmark test functions

Function		Bilevel-I	3-MOFBVE-I	4-MOFBVE-I	5-MOFBVE-I	SaNSDE
f_4	Mean	7.14e+09‡	5.55e+09‡	5.27e+09$^{\approx}$	5.92e+09‡	8.23e+09
	Std	3.58e+09	2.68e+09	2.41e+09	2.14e+09	3.09e+09
f_5	Mean	2.88e+06$^{\approx}$	2.96e+06$^{\approx}$	3.08e+06$^{\approx}$	3.15e+06$^{\approx}$	3.01e+06
	Std	5.50e+05	4.76e+05	7.52e+05	4.87e+05	4.25e+05
f_6	Mean	7.77e+04$^{\approx}$	8.56e+04$^{\approx}$	8.61e+04$^{\approx}$	9.16e+04$^{\approx}$	9.31e+04
	Std	3.13e+04	2.10e+04	1.60e+04	1.63e+04	1.49e+04
f_7	Mean	5.00e+06‡	3.95e+06‡	3.54e+06‡	4.30e+06‡	6.12e+06
	Std	2.75e+06	2.10e+06	9.68e+05	1.08e+06	2.65e+06
f_8	Mean	1.66e+13$^{\approx}$	2.15e+13$^{\approx}$	1.99e+13$^{\approx}$	1.94e+13$^{\approx}$	2.23e+13
	Std	6.38e+12	1.28e+13	8.26e+12	7.84e+12	1.50e+13
f_9	Mean	2.17e+08‡	2.04e+08‡	2.02e+08‡	2.69e+08$^{\approx}$	2.72e+08
	Std	3.10e+07	3.33e+07	3.13e+07	2.70e+07	2.49e+07
f_{10}	Mean	4.32e+03‡	2.35e+03‡	3.42e+03‡	4.26e+04$^{\approx}$	4.42e+04
	Std	9.05e+03	4.76e+02	9.19e+03	1.51e+04	1.25e+04
f_{11}	Mean	2.75e+08‡	2.26e+08‡	2.20e+08‡	3.11e+08$^{\approx}$	3.90e+08
	Std	8.47e+07	1.05e+08	6.67e+07	9.39e+07	2.01e+08
f_{13}	Mean	5.01e+08$^{\approx}$	4.72e+08$^{\approx}$	5.16e+08$^{\approx}$	5.15e+08$^{\approx}$	5.19e+08
	Std	2.09e+08	2.45e+08	1.61e+08	1.86e+08	2.83e+08
f_{14}	Mean	2.81e+08$^{\approx}$	5.42e+08$^{\approx}$	5.37e+08$^{\approx}$	9.67e+08$^{\approx}$	4.34e+08
	Std	2.45e+08	4.64e+08	7.43e+08	1.51e+09	4.63e+08
	w/t/l	5/5/0	5/5/0	4/6/0	2/8/0	–

Symbols '†', '‡', and '≈' denote the compared algorithms are worse than, better than, or similar to SaNSDE, respectively

nificantly. The performance of MOFBVE was evaluated on two different modified CEC-2010 and the CEC-2013 LSGO benchmark functions. The experimental results confirmed that MOFBVE with the specific feature, the transformation of the search space by reducing all variables into the important variables, has potential to play a positive role in solving LSGO problems. Furthermore, MOFBVE was compared with the standard CC algorithms, contribution-based CC algorithms, and SaNSDE which is the subcomponent optimizer in the CC algorithms and MOFBVE. The performance of MOFBVE is superior to or at least comparable with the other competitors. However, Omidvar et al. tried to detect the effect of the constructed subcomponents by decomposition method to improve CC algorithms in Omidvar et al. (2011, 2014a), the effect of each variable in an LSGO problem and their impact on LSGO algorithms have not directly considered. We made a basic attempt to show that detecting variables' effect is an important feature of LSGO problems to accelerate LSGO algorithms. This study is a starting idea in this direction to confirm the potentials and motivate other researchers in the LSGO fields to participate with the concept of variables' effect on the objective function. In future, we are planning to design hybrid strategies for MOFBVE with the CC algorithms to benefit from two features: interaction and various effects of variables. Furthermore, we will investigate the sensitivity of MOFBVE to the different levels of imbalance. Finally, we are interested in developing an iterative MOFBVE such that multilevel framework can use levels with most important variables not only at the beginning steps of optimization algorithm but also at other steps.

Acknowledgments The authors would like to thank anonymous reviewers for their constructive comments.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Appendix A

The the normal coefficient

Tables 22, 23, and 24 present the normal coefficient corresponding to nonseparable subcomponents in the modified normal CEC-2010 test functions.

Table 22 The normal coefficients for single-group m-nonseparable functions (f_4–f_8)

Function	f_4	f_5	f_6	f_7	f_8
Group1	7.78e+04	1.42e+02	3.86e+02	2.93e+01	1.27e+03

Table 23 The normal coefficients for $\frac{n}{2m}$-group m-nonseparable functions (f_9–f_{13})

Function	f_9	f_{10}	f_{11}	f_{12}	f_{13}
Group1	6.05e+01	3.96e+00	6.11e+01	1.89e+00	3.12e−01
Group2	2.77e−07	2.73e−05	1.38e+03	1.55e+05	2.21e−01
Group3	1.00e−04	6.70e−03	2.55e−01	1.49e−02	4.21e+01
Group4	4.75e−05	1.66e−02	9.62e+00	6.15e−03	1.47e+01
Group5	1.61e+01	1.60e+01	1.93e−01	5.63e−06	1.79e+02
Group6	2.57e+04	6.71e+02	4.88e+00	5.39e+02	2.16e+05
Group7	1.05e−01	7.97e+00	2.09e+01	3.99e+02	4.67e+03
Group8	2.74e+02	7.60e−02	1.41e−02	5.76e−01	1.41e−04
Group9	4.33e+01	2.80e+02	6.68e+00	4.96e+02	1.03e−07
Group10	7.00e−04	2.49e+02	6.36e+01	3.56e+00	5.08e+02

Table 24 The normal coefficients for $\frac{n}{m}$-group m-nonseparable functions (f_{14}–f_{18})

Function	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}
Group1	3.11e−06	7.13e+02	2.83e+02	1.14e−02	2.99e−01
Group2	1.59e+00	5.06e−02	2.95e+01	7.11e+07	3.61e−02
Group3	1.28e+00	8.85e+01	1.98e+02	4.50e+01	3.24e+02
Group4	3.48e+07	8.31e−02	2.16e+02	7.63e+00	4.12e+07
Group5	6.20e−01	1.31e+02	3.62e−05	4.64e−03	1.07e−04
Group6	4.00e+01	1.77e+04	4.18e+01	6.39e−04	2.43e+00
Group7	5.10e+00	1.54e−05	5.31e−01	4.95e−06	4.71e−05
Group8	5.46e+00	1.22e+03	5.24e−03	5.39e−02	8.08e+03
Group9	1.62e+00	2.37e+04	8.30e−03	6.93e−04	1.70e+04

Table 24 continued

Function	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}
Group10	1.49e−02	1.39e+00	6.99e+03	8.78e+01	1.03e−05
Group11	2.15e−04	1.73e+05	3.72e−03	1.11e−01	6.78e+05
Group12	8.90e+00	2.93e+00	1.95e−04	2.03e+05	5.57e−04
Group13	9.36e−05	1.94e−04	4.41e+00	3.40e+04	4.79e+00
Group14	8.01e−04	2.63e−07	1.08e+06	3.10e+00	1.98e+03
Group15	9.85e+03	9.99e−02	1.19e+00	1.42e−01	2.76e+00
Group16	5.54e−02	1.38e+02	8.41e+00	2.86e+03	7.71e+06
Group17	3.79e−01	8.96e+00	1.53e−03	3.63e−04	1.81e+08
Group18	5.01e+02	1.74e+01	1.05e+05	1.05e+02	2.60e+00
Group19	1.26e−01	1.86e−02	2.37e+00	9.83e−03	1.90e−06
Group20	1.22e+03	2.70e+00	3.89e+01	6.30e−02	8.04e−02

Appendix B

The benchmark functions

– **CEC-2010 benchmark functions**

Dimension: $D = 1000$
Group size: $m = 50$
$x = (x_1, x_2, \ldots, x_D)$: The candidate solution
$o = (o_1, o_2, \ldots, o_D)$: The (shifted) global optimum
$z = x - o, z = (z_1, z_2, \ldots, z_D)$: The shifted candidate solution
P: A random permutation of $1, 2, \ldots, D$

$$F_{\text{elliptic}}(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} {x_i}^2$$

$$F_{\text{rosenbrock}} = \sum_{i=1}^{D-1} [100({x_i}^2 - x_{i+1})^2 + (x_i - 1)^2]$$

$$F_{\text{rastrigin}} = \sum_{i=1}^{D} [{x_i}^2 - 10\cos(2\pi) x_i + 10]$$

$$F_{\text{ackley}} = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} {x_i}^2}\right) - \exp\left(\frac{1}{D}\right)\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + c$$

$$F_{\text{schwefel}} = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_i\right)^2$$

$$F_9(x) = \sum_{k=1}^{\frac{D}{2m}} F_{\text{rot\_elliptic}}[z(P_{(k-1)*m+1} : P_{k*m})] + F_{\text{rot}_{\text{elliptic}}}[z(P_{\frac{D}{2}+1} : P_D)]$$

$$F_{10}(x) = \sum_{k=1}^{\frac{D}{2m}} F_{\text{rot\_rastrigin}}[z(P_{(k-1)*m+1} : P_{k*m})] + F_{\text{rastrigin}}[z(P_{\frac{D}{2}+1} : P_D)]$$

$$F_{11}(x) = \sum_{k=1}^{\frac{D}{2m}} F_{\text{rot\_ackley}}[z(P_{(k-1)*m+1} : P_{k*m})] + F_{\text{ackley}}[z(P_{\frac{D}{2}+1} : P_D)]$$

$$F_{12}(x) = \sum_{k=1}^{\frac{D}{2m}} F_{\text{schwefel}}[z(P_{(k-1)*m+1} : P_{k*m})] + F_{\text{sphere}}[z(P_{\frac{D}{2}+1} : P_D)]$$

$$F_{13}(x) = \sum_{k=1}^{\frac{D}{2m}} F_{\text{rot\_rosenbrock}}[z(P_{(k-1)*m+1} : P_{k*m})] + F_{\text{sphere}}[z(P_{\frac{D}{2}+1} : P_D)]$$

$$F_{14}(x) = \sum_{k=1}^{\frac{D}{m}} F_{\text{rot\_elliptic}}[z(P_{(k-1)*m+1} : P_{k*m})]$$

$$F_{15}(x) = \sum_{k=1}^{\frac{D}{m}} F_{\text{rot\_rastrigin}}[z(P_{(k-1)*m+1} : P_{k*m})]$$

$$F_{16}(x) = \sum_{k=1}^{\frac{D}{m}} F_{\text{rot\_ackley}}[z(P_{(k-1)*m+1} : P_{k*m})]$$

$$F_{17}(x) = \sum_{k=1}^{\frac{D}{m}} F_{\text{schwefel}}[z(P_{(k-1)*m+1} : P_{k*m})]$$

$$F_{18}(x) = \sum_{k=1}^{\frac{D}{m}} F_{\text{rosenbrock}}[z(P_{(k-1)*m+1} : P_{k*m})]$$

– **CEC-2013 benchmark functions**

Dimension: $D = 1000$
Group size: $m = 50$

$S = 50, 25, 25, 100, 50, 25, 25, 700$

$S1 = \{50, 50, 25, 25, 100, 100, 25, 25, 50, 25, 100, 25, 100, 50, 25, 25, 25, 100, 50, 25\}$

x^{opt} : The optimum decision vector
P: A random permutation of $1, 2, \ldots, D$
T_{osz}: A transformation function to create smooth local irregularities.
T_{asy}: A transformation function to break the symmetry of the symmetric functions.
λ: A D-dimensional diagonal matrix with the diagonal elements is used to create ill-conditioning.
R: An orthogonal rotation matrix which is used to rotate the fitness landscape randomly around various axes
m: The overlap size between subcomponents

$$y = x - x^{\text{opt}}$$

$$y_i = y(P_{[C_{i-1}+1]} : P_{[C_i]}) \quad i \in 1, \ldots, |S|,$$

$$y_{i1} = y(P_{[C_{i-1}-(i-1)m+1]} : P_{[C_i-(i-1)m]}) \quad i \in 1, \ldots, |S|,$$

$$y_{i2} = y(P_{[C_{i-1}-(i-1)m+1]} : P_{[C_i-(i-1)m]}) - x_i^{\text{opt}} \quad i \in 1, \ldots, |S|,$$

$$z_i = T_{\text{osz}}(R_i y_i), \quad i \in 1, \ldots, |S| - 1$$

$$z_{i1} = T_{\text{asy}}^{0.2} T_{\text{osz}}(R_i y_{i1}), \quad i \in 1, \ldots, |S| - 1$$

$$z_{i2} = T_{\text{asy}}^{0.2} T_{\text{osz}}(R_i y_{i2}), \quad i \in 1, \ldots, |S| - 1$$

$$z_{|S|} = T_{\text{osz}}(y_{|S|})$$

$$R_i : a|S_i| \times |S_i|$$

$$F_4(x) = \sum_{k=1}^{|s|-1} w_i f_{\text{elliptic}}(z_i) + f_{\text{elliptic}}(z_{|s|}),$$

$$F_5(x) = \sum_{k=1}^{|s|-1} w_i f_{\text{rastrigin}}(z_i) + f_{\text{rastrigin}}(z_{|s|}),$$

$$F_6(x) = \sum_{k=1}^{|s|-1} w_i f_{\text{ackley}}(z_i) + f_{\text{ackley}}(z_{|s|}),$$

$$F_7(x) = \sum_{k=1}^{|s|-1} w_i f_{\text{schwefel}}(z_i) + f_{\text{schwefel}}(z_{|s|}),$$

$$F_8(x) = \sum_{k=1}^{|s1|} w_i f_{\text{elliptic}}(z_i),$$

$$F_9(x) = \sum_{k=1}^{|s1|} w_i f_{\text{rastrigin}}(z_i),$$

$$F_{10}(x) = \sum_{k=1}^{|s1|} w_i f_{\text{ackley}}(z_i),$$

$$F_{11}(x) = \sum_{k=1}^{|s1|} w_i f_{\text{schwefel}}(z_i),$$

$$F_{13}(x) = \sum_{k=1}^{|s1|} w_i f_{\text{schwefel}}(z_{i1}),$$

$$F_{14}(x) = \sum_{k=1}^{|s1|} w_i f_{\text{schwefel}}(z_{i2}),$$

References

Andrea S, Karen C, Marian Scott E et al (2000) Sensitivity analysis, vol 134. Wiley, New York

Andrea S, Marco R, Terry A, Francesca C, Jessica C, Debora G, Michaela S, Stefano T (2008) Global sensitivity analysis: the primer. Wiley, New York

Benjamin D, Dirk S, Carsten W (2013) When do evolutionary algorithms optimize separable functions in parallel? In: Proceedings of the twelfth workshop on Foundations of genetic algorithms XII, pp 51–64. ACM

Daniel M, Manuel L, Francisco H (2010) Ma-sw-chains: Memetic algorithm based on local search chains for large scale continuous global optimization. In: Evolutionary Computation (CEC), 2010 IEEE Congress on, IEEE, pp 1–8

Eman S, Daryl E, Ruhul S (2012a) Dependency identification technique for large scale optimization problems. In: Evolutionary Computation (CEC), 2012 IEEE Congress on, IEEE, pp 1–8

Eman S, Daryl E, Ruhul S (2012b) Using hybrid dependency identification with a memetic algorithm for large scale optimization problems. In: Simulated evolution and learning, Springer, pp 168–177

Francesca C, Jessica C, Andrea S (2007) An effective screening design for sensitivity analysis of large models. Environ Model Softw 22(10):1509–1518

Frank W (1945) Individual comparisons by ranking methods. Biometr Bull, pp 80–83

Frans Van den B, Andries PE (2004) A cooperative approach to particle swarm optimization. Evol Comput IEEE Trans 8(3):225–239

Hanning C, Yunlong Z, Kunyuan H, Xiaoxian H, Ben N (2008) Cooperative approaches to bacterial foraging optimization. In: Advanced intelligent computing theories and applications. with aspects of artificial intelligence, Springer, NY, pp 541–548

Hemant Kumar S, Tapabrata R (2010) Divide and conquer in coevolution: A difficult balancing act. In: Agent-based evolutionary search, Springer, pp 117–138

Herschel R, Ömer FA (1999) General foundations of high-dimensional model representations. J Math Chem 25(2–3):197–233

Hui W, Shahryar R, Zhijian W (2013) Parallel differential evolution with self-adapting control parameters and generalized opposition-based learning for solving high-dimensional optimization problems. J Parallel Distrib Comput 73(1):62–73

Jasbir A (2004) Introduction to optimum design. Academic Press, New York

Jinpeng L, Ke T (2013) Scaling up covariance matrix adaptation evolution strategy using cooperative coevolution. In: Intelligent data engineering and automated learning–IDEAL 2013, Springer, pp 350–357

Joaquín D, Salvador G, Daniel M, Francisco H (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1(1):3–18

Ke T, Xiaodong L, Ponnuthurai Nagaratnam S, Zhenyu Y, Thomas W (2010) Benchmark functions for the CEC'2010 special session and competition on large-scale global optimization. Technical report, Nature Inspired Computation and Applications Laboratory (NICAL),USTC, China. http://www.it-weise.de/documents/files/TLSYW2009BFFTCSSACOLSGO.pdf

Liang S, Shinichi Y, Xiaochun C, Yanchun L (2012) A cooperative particle swarm optimizer with statistical variable interdependence learning. Inf Sci 186(1):20–39

MacQueen J et al (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability., vol 1. California, pp 281–297

Max DM (1991) Factorial sampling plans for preliminary computational experiments. Technometrics 33(2):161–174

Mitchell AP, de Kenneth AD (1994) A cooperative coevolutionary approach to function optimization. In: Parallel problem solving from naturePPSN III, Springer, pp 249–257

Mitchell AP (1997) The design and analysis of a computational model of cooperative coevolution. PhD thesis, Citeseer

Mohammad Nabi O, Xiaodong L, Ke T (2015) Designing benchmark problems for large-scale continuous optimization. Inf Sci 316:419–436

Mohammad Nabi O, Xiaodong L, Xin Y (2011) Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms. In: Proceedings of the 13th annual conference on genetic and evolutionary computation, ACM, pp 1115–1122

Mohammad Nabi O, Xiaodong L, Yi M, Xin Y (2014a) Cooperative co-evolution with differential grouping for large scale optimization. Evolutionary Computation, IEEE Transactions on 18(3): 378–393

Mohammad Nabi O, Yi M, Xiaodong L (2014b) Effective decomposition of large-scale separable continuous functions for cooperative co-evolutionary algorithms. In: Evolutionary computation (CEC), 2014 IEEE Congress on, IEEE, pp 1305–1312

Sedigheh M, Mohammad Ebrahim S, Shahryar R (2015) Metaheuristics in large-scale global continues optimization: a survey. Inf Sci 295:407–428

Sedigheh M, Mohammad Ebrahim S, Shahryar R (2014) Cooperative co-evolution with a new decomposition method for large-scale optimization. In: Evolutionary Computation (CEC), 2014 IEEE Congress on, IEEE, pp 1285–1292

Singiresu SR, Rao SS (2009) Engineering optimization: theory and practice. Wiley, New York

Wenxiang C, Thomas W, Zhenyu Y, Ke T (2010) Large-scale global optimization using cooperative coevolution with variable interaction learning. In: Parallel Problem Solving from Nature, PPSN XI, Springer, Heidelberg, pp 300–309

Xiaodong L, Ke T, Mohammad NO, Zhenyu Y, Kai Q (2013) Benchmark functions for the cec 2013 special session and competition on large-scale global optimization. Gene 7(33):8

Xiaodong L, Xin Y (2009) Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms. In: Evolutionary computation, 2009. CEC'09. IEEE Congress on, pp 1546–1553. IEEE

Xiaodong L, Xin Y (2012) Cooperatively coevolving particle swarms for large scale optimization. Evol Comput IEEE Trans 16(2):210–224

Yoel T, Chi-Keong G (2010) Computational intelligence in expensive optimization problems, vol 2. Springer, New York

Yong L, Xin Y, Qiangfu Z, Tetsuya H (2001) Scaling up fast evolutionary programming with cooperative coevolution. In: Evolutionary Computation, 2001. Proceedings of the 2001 Congress on, IEEE. vol 2, pp 1101–1108

Yuanfang Ren, Yan Wu (2013) An efficient algorithm for high-dimensional function optimization. Soft Comput 17(6):995–1004

Zhao S-Z, Ponnuthurai Nagaratnam S, Swagatam D (2011) Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. Soft Comput 15(11):2175–2185

Zhenyu Y, Ke T, Xin Y (2008a) Large scale evolutionary optimization using cooperative coevolution. Inf Sci 178(15):2985–2999

Zhenyu Y, Ke T, Xin Y (2008b) Multilevel cooperative coevolution for large scale optimization. In: Evolutionary computation, 2008. CEC 2008. IEEE World Congress on Computational Intelligence). IEEE Congress on, IEEE, pp 1663–1670

Zhenyu Y, Ke T, Xin Y (2008c) Self-adaptive differential evolution with neighborhood search. In: Evolutionary computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on, IEEE, pp 1110–1116