Appl Intell
DOI 10.1007/s10489-017-0926-z

Cooperative co-evolution with sensitivity analysis-based budget assignment strategy for large-scale global optimization

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

Abstract Cooperative co-evolution has proven to be a successful approach for solving large-scale global optimization (LSGO) problems. These algorithms decompose the LSGO problems into several smaller subcomponents using a decomposition method, and each subcomponent of the variables is optimized by a certain optimizer. They use a simple technique, the round-robin method, to equally assign the computational time. Since the standard cooperative co-evolution algorithms allocate the computational budget equally, the performance of these algorithms deteriorates for solving LSGO problems with subcomponents by various effects on the objective function. For this reason, it could be very useful to detect the subcomponents' effects on the objective function in LSGO problems. Sensitivity analysis methods can be employed to identify the most significant variables of a model. In this paper, we propose a cooperative co-evolution algorithm with a sensitivity analysis-based budget assignment method (SACC), which can allocate the computational time among all subcomponents based on their different effects on the objective function, accordingly. SACC is benchmarked on imbalanced LSGO problems. Simulation results confirm that SACC obtains a promising performance on the majority of the imbalanced LSGO benchmark functions.

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

1 Introduction

Many real-world problems can be formulated as optimization problems having a large number of decision variables [11, 15, 35]. A large number of metaheuristic algorithms have been applied to solve real-world problems, but the performance of these algorithms deteriorates when the dimension of the optimization problem increases [2, 11, 36, 38]. There are two main reasons for the low performance of metaheuristic algorithms [2, 11, 36, 38]. First, their landscape complexity increases with the increase in the problem dimension. Second, the search space exponentially increases with the problem size. Potter and De Jong in 1994 [23, 24] proposed the first cooperative co-evolution (CC) framework to solve n-dimensional optimization problems. These methods used a divide-and-conquer strategy to decompose an n-dimensional decision vector into some low-dimensional subcomponents and then evolved these subcomponents separately. CC algorithms lose their efficiency on non-separable problems when interacting variables are placed in different subcomponents [24]. Thus, subcomponents cannot be efficiently evolved independently, since the influence of a variable on the fitness value in one subcomponent depends on other variables in different subcomponents. In CC algorithms, the ideal decomposition method should associate variables into subcomponents

✉ Shahryar Rahnamayan
Shahryar.Rahnamayan@uoit.ca

Sedigheh Mahdavi
Sedigheh.Mahdavi@uoit.ca

Mohammad Ebrahim Shiri
shiri@aut.ac.ir

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

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

Published online: 02 May 2017

such that the interdependencies among different subcomponents are minimal [18, 39]. A major part of the challenge of CC algorithms is the design of a decomposition method that is able to minimize the interdependencies among different subcomponents. Recently, a large number of CC algorithm modifications have been developed to enhance the detection of interacting variables. These algorithms can be divided into two major categories in terms of the utilized decomposition strategy, namely, static and dynamic decomposition methods [15]. In the static grouping based CC methods, a fixed value 'k' is selected for the subcomponent size and then the associated variables of each subcomponent remain fixed in all the optimization iterations. In the dynamic CC methods, the knowledge of the subcomponent structure is dynamically obtained either before or during the optimization process.

In addition, another significant feature of CC algorithms is the computational budget assignment among all subcomponents. This plays a critical role in the performance improvement in the CC framework to tackle imbalanced LSGO problems. Most real-world optimization problems have some imbalance among different subcomponents, i.e., their contributions to the objective function are different [11, 18]. The performance of the standard CC algorithms deteriorate on the imbalanced LSGO problems, because they divide computational resources equally among subcomponents (i.e., round-robin method) [18, 22]. Omidvar et al. [18, 22] proposed a contribution based cooperative co-evolution (CBCC) method, in which the subcomponent having the maximum effect on the objective function is selected for further optimization. They introduced two versions of the CBCC method, CBCC1 and CBCC2. CBCC1 optimizes the selected subcomponent for only one iteration, whereas CBCC2 optimizes it until the fitness value is improved. They indicated that with an accurate decomposition method, CBCC methods can accurately compute the contribution of the subcomponents. The CBCC methods spend more computational budget on only the one subcomponent that has the maximum effect. By considering the effect of all the subcomponents, it is possible to devise a better budget assignment method. Only a limited number of research works have been developed to study the different effects of subcomponents on the objective function. Thus, it would be interesting to investigate how CC algorithms assign computational resources among all the subcomponents to improve their performance [15, 19]. Obviously, a good budget scheduling method can improve the performance of CC algorithms for solving imbalanced LSGO problems.

In this paper, we propose a CC algorithm with a sensitivity analysis-based budget assignment method (called SACC), where it can assign a specific amount of the computational budget to each subcomponent according to its effect on the objective function. The sensitivity analysis technique is an important tool for studying the influence of the variation in the models' input parameters on the variation in its outputs [25, 27, 28]. In addition, screening methods are a type of sensitivity analysis methods that are applied to compute the sensitivity of the input parameters in a model. The main effect for each subcomponent is computed by using the Morris screening method. Then, the number of the optimization iterations for each subcomponent is determined according to its computed main effect. The performance of the SACC is evaluated on a set of benchmark functions, which are modified types of CEC-2010 benchmark problems having the imbalanced subcomponents and CEC-2013 LSGO benchmark functions. The proposed method is compared with the standard CC and contribution based CC algorithms. Furthermore, our results confirm that the SACC performs significantly better than other methods on the imbalanced LSGO problems.

The organization of the rest of this paper is as follows. Section 2 gives a background review of cooperative co-evolution algorithms, various decomposition methods, and the Morris screening method. In Section 3, the proposed CC framework is described in detail. In Section 4, the experimental results and discussion are presented. Finally, the paper is concluded in Section 5.

2 Background review

2.1 Cooperative co-evolution

In [23, 24], standard CC algorithms were introduced for decomposing LSGO problems based on a one-dimensional based strategy. The main steps of the CC framework are described as follows:

1. Decompose a high-dimensional problem into m low-dimensional non-overlapping subcomponents.
2. Set $k = 1$ to begin a loop, which is repeated for the number of subcomponents.
3. Evolve each subcomponent by an optimization algorithm for a few iterations in the round-robin scheme.
4. If $k < m$, then $k++$, and go to Step 3.
5. Stop the evolutionary process if the termination condition is satisfied; otherwise, go to Step 2.

For computing the fitness of an individual, an n-dimensional vector is constructed using this individual and the selected members from other subcomponents. Liu et al. [13] proposed a CC framework with Fast Evolutionary Programming (FEP), called FEPCC, to solve optimization problems with up to 1000 dimensions. In 2004, Van den Bergh and Engelbrecht applied the CC framework using PSO algorithms in which an n-dimensional problem is partitioned

into K s-dimensional sub-problems ($s << n$) [37]. In [33], a splitting-in-half strategy was proposed in which an n dimensional problem is divided into two $n/2$ subcomponents. In [39], a random decomposition method (DECC-G) was introduced for handling large scale non-separable problems with up to 1000 dimensions. In DECC-G, an LSGO problem is randomly decomposed into several smaller subcomponents. DECC-G increases the probability that interaction variables are assigned to one same subcomponent. A major drawback of the DECC-G method is that its performance is degraded when the number of interacting variables increases [20]. In the mentioned CC methods, a major difficulty is the detection of the appropriate value for the subcomponent size. Yang et al. [40] proposed a multilevel cooperative co-evolution (MLCC) algorithm which uses a decomposer pool based on a set of possible subcomponent sizes. The MLCC method assigns some selection probabilities to all decomposers, which are adaptively updated according to the performance of each decomposer in the optimization process. In this method, the selection of the subcomponent size is difficult. Also, after a subcomponent size is selected, the problem is decomposed into a set of subcomponents with the equal size [18].

Ray and Yao [26, 34] introduced the correlation matrix-based algorithms in which the subcomponents are constructed according to the computed correlations of the variables in each generation. These methods have a low performance for the detection of the nonlinear dependencies among variables and also use vast computational resources [17]. In [21], a delta decomposition method was proposed for decomposing LSGO problems based on the absolute amount of change in each dimension at two sequential cycles. The performance of the delta method decreases to solve non-separable problems having multiple non-separable subcomponents [18]. A CC method with variable interaction learning (CCVIL) was introduced in [4] that can adaptively discover the interaction among variables. The CCVIL method has two stages, learning and optimization. In the learning stage, each variable is placed in a subcomponent and optimized similar to a basic CC methods at the limited cycles. After each subcomponent is optimized, the interaction among current and previous optimized subcomponents is considered for merging interdependent subcomponents and then new subcomponents are optimized in the optimization stage. The challenging part of CCVIL is determining an appropriate balance between the fitness evaluations in the learning and optimization stages [3].

Sayed et al. [29, 30] proposed a dependency identification (DI) algorithm which uses an internal minimization problem to create subcomponents according to the concept of partially separable functions during the CC process. In [10], an adaptive cooperative particle swarm optimizer was proposed based on using learning automata. Action set and the action probability vectors of learning automata were updated based on the population to identify the interaction variables and integrate them into a joint swarm. Mahdavi et al. [14] introduced a decomposition algorithm (DM-HDMR) based on the high dimensional model representation method. In DM-HDMR, first an RBF-HDMR model is approximated by using the first-order RBF-HDMR proposed by Shan and Wang [31]. Then, the interactions among variables are recognized according to the obtained correlations of the first-order RBF-HDMR to create subcomponents. Liu and Tang [12] proposed the cooperative coevolution covariance matrix adaptation evolution strategy (CC-CMA-ES), which uses a CC framework with the CMA-ES algorithm for handling with LSGO problems. They introduced two new decomposition methods according to the diagonal of the covariance matrix of the CMA-ES algorithm. It has been shown recently [11, 18] that, in the presence of a highly accurate decomposition method, budget scheduling methods are beneficial. Therefore, we selected a recent decomposition method, namely, the differential decomposition method which has high accuracy on the balanced CEC-2010 benchmark functions.

2.2 Differential grouping (DG) method

In [18], a differential grouping (DG) method was proposed based on the definition of the partial separable functions. They defined and proved a theorem to identify pairwise interacting variables, which is described as follows.

Theorem 1 *If the following condition holds for an additively separable function* $f(\mathbf{x})$*:*

$$\forall a, b_1 \neq b_2, \delta \in \mathbb{R}, \delta \neq 0, \Delta_{\delta,x_p}[f](\mathbf{x})|_{x_p=a,x_q=b_1} \neq \Delta_{\delta,x_p}[f](\mathbf{x})|_{x_p=a,x_q=b_2} \quad (1)$$

then x_p *and* x_q *are non-separable, where* $\Delta_{(\delta,x_p)}[f](\mathbf{x}) = f(\ldots, x_p + \delta, \ldots) - f(\ldots, x_p, \ldots)$ *is the forward difference of* f *according to variable* x_p *with the interval* δ*. In the DG algorithm, first, for an array including all the variables, the interactions among all variables with the first variable of the function are investigated based on this theorem. Then, the variables having an interaction with the first variable are placed in the same subcomponent and removed from the array of variables. This process is repeated for all the remaining variables in the array. Variables that do not have any interaction with other variables are assigned to a separable subcomponent. When DG constructs non-separable and separable subcomponents, the standard CC framework* [23, 24] *is applied to optimize the subcomponents in a round-robin fashion by using a self-adaptive evolution differential with the neighbourhood search strategy (SaNSDE)* [41].

2.3 Morris screening method

The models of real-world processes are computationally expensive and involve a large number of input factors. Screening methods can identify the most important input factors among a set of input factors with a few model evaluations. In 1991, Morris [16] proposed a screening method to identify the input factors with the strongest effect on the output variability in models with many input factors. The steps of Morris method are as follows.

1. The range of each model input factor $X_i, i = 1, \ldots, n$ is divided into p levels in the space of the input factors (n-dimensional p-level grid).
2. For each factor, a base point x is randomly chosen on the p-level grid.
3. An elementary effect (EE_i) for each input factor i is defined by:

$$EE_i(\mathbf{x}) = \frac{f(x_1, \ldots, x_{i-1}, x_i + \Delta, \ldots, x_n) - f(\mathbf{x})}{\Delta}, \tag{2}$$

   where Δ is a multiple of $1/(k-1)$ and p is the number of levels.
4. Several values of EE_i are computed to provide measures of sensitivity. Therefore, for computing the expected values of EE_i r trajectories are generated. Each trajectory includes $n+1$ points, changing each input factor exactly once at different locations in the level grid.
5. The mean and the standard deviation distribution of r EE_i are computed as two sensitivity measures, μ and σ.

In the Morris method, the number of evaluations is $r(n+1)$. 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. [1] proposed the mean (μ^*) of the absolute elementary values instead of the μ value. Also, they suggested selecting $r = [10, 20]$, $p = [4, 8]$, and $\Delta = p/2(p-1)$ which most screening method research studies have used these suggested values. In this paper, we employed the mean (μ^*) of the absolute elementary values and $r = 20$, $p = 8$, and $\Delta = p/2(p-1)$. The source code of the Morris method can be found in [7].

3 Cooperative co-evolution algorithm with sensitivity analysis-based budget assignment method

The standard CC algorithms use the round-robin method to optimize all the subcomponents; therefore, some fraction of the computational budget may be wasted by some subcomponents that contribute little to the global fitness [18, 22]. Many real-world problems have some non-separable subcomponents, which have various effects on the objective function. Therefore, a significant challenge of CC algorithms is how the computational budget can be divided among all subcomponents according to their effects. In this section, we propose using the Morris screening method to divide the computational budget among various subcomponents according to their main effect on the global fitness value. In the following, the three strategies of sensitivity analysis-based Cooperative Co-evolution (SACC1, SACC2, and SACC3) methods are described in detail. Algorithm 1 shows the details of the proposed strategies. First, the subcomponents are constructed via a decomposition method (line 2). Then, the Morris method is employed to compute the main effect (μ^*) of each variable (line 4). A normalized ratio ME_s, called the main effect, is computed for each subcomponent proportional to its effect on the fitness value, which is defined as the sum effects of all its variables divided by the sum effects of all the decision variables (line 6).

The main effect of a subcomponent s is calculated based on Morris measure μ^* as follows:

$$ME_s = \frac{\sum_{i=1}^{k} \mu_i^\star}{\sum_{i=1}^{n} \mu_i^\star}, \tag{3}$$

where parameters k and n are the number of variables in the subcomponent s and all variables of the function, respectively; and parameter $\mu_i^\star$ is the Morris sensitivity measure for the variable i. ME_s provides an ordering of the all subcomponents' effects to select the subcomponent having the maximum effect. In SACC1 (lines 10–38), the subcomponent with the maximum ME_s is selected which this subcomponent has the most effect on the global fitness value. Then, the selected subcomponent is optimized at only one iteration (lines 25–35) after that all subcomponents are optimized in a round-robin fashion at each cycle of the CC algorithm. The SACC2 (lines 10–38) strategy is similar to SACC1, but it optimizes the subcomponent with the maximum ME_s by the different number of iteration. In fact, after optimizing the subcomponent with the maximum ME_s, it computes the difference among the current best obtained solution and the previous one (line 32) as the improvement of the best obtained solution. Then, this subcomponent is optimized until it can improve the best obtained solution (lines 25–37).

The SACC3 strategy (lines 39–50) assigns a special number of the computational budgets (line 7), $Iter_s$, for a subcomponent s which is defined as:

$$Iter_s = c_1 + \lfloor \log\left(\frac{ME_s}{ME_{min}}\right) \rfloor, \tag{4}$$

Algorithm 1 (best,best_val) $\Leftarrow$ **SACC**($func$, n, max_cycle, strategy)

$pop \Leftarrow$ **initialize**($popsize, n$)
($groups, num\_groups$) $\Leftarrow$ **decomposition method**($func, n$) //construct subcomponents.
($best, best\_val$) $\Leftarrow$ **evaluate**(pop)
(μ, σ) $\Leftarrow$ **Morris**(n) //computing Morris sensitivity measures.
for $s = 1$ to num_groups **do**
    Calculate the main effect, i.e., $ME_s = \frac{\sum_{i=1}^{k} \mu_i^\star}{\sum_{i=1}^{n} \mu_i^\star}$, for each subcomponent s
//computing $Iter_s$ for SACC3 strategy.
    Calculate the number of iteration, i.e., $Iter_s = c_1 + \lfloor \log\left(\frac{ME_s}{ME_{min}}\right) \rfloor$, for each subcomponent s
end for
[$Max\_subcomponent, Maxsubcomponent\_index$] $\Leftarrow max(ME_s)$
if $strategy =$ "$SACC1$" || $strategy =$ "$SACC2$" **then**
    $i = 1$
    while $i \neq max\_cycle$ **do**
        for $j = 1$ to num_groups **do**
            $index\_subpop \Leftarrow groups[j]$
            $subpop \Leftarrow pop(:, index\_subpop)$
            $Iteropt = 1$
            $subpop \Leftarrow$ **optimizer**($best, subpop, Iteropt$)
            $pop(:, index\_subpop) \Leftarrow subpop$
            ($best, best\_val$) $\Leftarrow$ **evaluate**(pop)
            $i = i + Iteropt$
        end for
        $Diff = 1$
        while Diff$\neq$0 **do**
            $index\_subpop \Leftarrow groups[Maxsubcomponent\_index]$
            $subpop \Leftarrow pop(:, index\_subpop)$
            $Iteropt = 1$
// optimizing subcomponent with maximum effect.
            $subpop \Leftarrow$ **optimizer**($best, subpop, Iteropt$)
            $pop(:, index\_subpop) \Leftarrow subpop$
            $bestval\_old \Leftarrow best\_val$
            $pop(:, index\_subpop) \Leftarrow subpop$
            ($best, best\_val$) $\Leftarrow$ **evaluate**(pop)
            $Diff \Leftarrow abs(bestval\_old - best\_val)$ //abs(x) computes the absolute value of x.
            if $strategy =$ "$SACC1$" **then**
                $Diff \Leftarrow 0$ //it terminates loop for SACC1.
            end if
            $i = i + Iteropt$
        end while //SACC2 can continue loop until Diff$\neq$0.
    end while
else if $strategy =$ "$SACC3$" **then**
    $i = 1$
    while i $\neq$ max_cycle **do**
        for $j = 1$ to num_groups **do**
            $index\_subpop \Leftarrow groups[j]$
            $subpop \Leftarrow pop(:, index\_subpop)$
            $Iteropt = 1 + Iter_j$ //assigning budget ($Iter_j$) between subcomponents in SACC3.
//optimizing each subcomponent with iteration $Iteropt$.
            $subpop \Leftarrow$ **optimizer**($best, subpop, Iteropt$)
            $pop(:, index\_subpop) \Leftarrow subpop$
        end for
        $i = i + Iteropt$
    end while
end if

where ME_{min} is the minimum of the ME_s values for all subcomponents and c_1 is a predefined number of the computational budget to optimize a subcomponent in the standard CC algorithms. Also, $\lfloor . \rfloor$ is the floor function. Note that $\mu_i^\star$ value is related to the absolute value of the objective function and therefore $Iter_s$ cannot be obtained directly by dividing the absolute value $\mu_i^\star$ of the subcomponents. $Iter_s$ is constructed based on the ME_s value, which gives a ratio value for the effect of each subcomponent (i.e., subcomponents can be sorted according to their effects). $Iter_s$ is carefully formulated such that the subcomponents with the higher effects have a greater number of the computational budget and vice versa. In SACC3, the contribution of each subcomponent to the global fitness value can be estimated based on the sensitivity measure $\mu_i^\star$ in the Morris method. The SACC3 strategy attempts to divide the computational budget among different subcomponents based on their approximated contributions. On the other hand, if the effects of all the subcomponents are very similar and the Morris method can approximate ME_s values with high accuracy, $Iter_s$ is equal to 1. It is important to mention that the normalized ratio ME_s values are proportional to the effect of the subcomponents on the fitness value, and therefore, we can apply other methods that are able to compute these normalized ratio values. In CC algorithms, the number of the computational budget for each subcomponent is 1 and in this paper, c_1 was also set to 1. Note that a significant advantage of the proposed method is that it does not add any heuristic parameter to the basic CC algorithms. We consider some simple examples to demonstrate how the SACC3 strategy assigns $Iter_s$. In all the examples, we used the source code of the Morris method in [7] to compute $\mu_i^\star$ values.

Example 1 For $k = 4$ and $n = 16$, consider a non-separable function which is expressed as follows:

$$\begin{aligned} f(\mathbf{X}) = {} & 10^{-4}.f_{rastrigin}(x_1, x_2, x_3, x_4) \\ & + 10^{-2}.f_{rastrigin}(x_5, x_6, x_7, x_8) \\ & + 10^{2}.f_{rastrigin}(x_9, x_{10}, x_{11}, x_{12}) \\ & + 10^{4}.f_{rastrigin}(x_{13}, x_{14}, x_{15}, x_{16}) \end{aligned}$$

where $f_{rastrigin}(\mathbf{X}) = \sum_{i=1}^{D} {x_i}^2 - 10.\cos(2\pi x_i) + 10.$ This non-separable function has four subcomponents with the different effects. First, $\mu_i^\star$ values of 16 variables are computed by the Morris method. Table 1 shows $\mu_i^\star$ values. Then, ME_s values of four subcomponents are computed and we have:

$$ME_1 = \frac{\sum_{i=1}^{4} \mu_i^\star}{\sum_{i=1}^{16} \mu_i} = 9.68 \times 10^{-9},$$

$$ME_2 = \frac{\sum_{i=5}^{8} \mu_i^\star}{\sum_{i=1}^{16} \mu_i} = 9.93 \times 10^{-7},$$

$$ME_3 = \frac{\sum_{i=9}^{12} \mu_i^\star}{\sum_{i=1}^{16} \mu_i} = 9.38 \times 10^{-3},$$

$$ME_4 = \frac{\sum_{i=13}^{16} \mu_i^\star}{\sum_{i=1}^{16} \mu_i} = 9.91 \times 10^{-1},$$

Thus,

$$\begin{aligned} Iter_1 &= 1 + \lfloor \log\left(\frac{ME_1}{ME_{min}}\right) \rfloor = 1 + \lfloor \log((1) \rfloor = 1, \\ Iter_2 &= 1 + \lfloor \log\left(\frac{ME_2}{ME_{min}}\right) \rfloor = 1 + \lfloor \log\left(1.03 \times 10^2\right) \rfloor \\ &= 7, \end{aligned}$$

$$\begin{aligned} Iter_3 &= 1 + \lfloor \log\left(\frac{ME_3}{ME_{min}}\right) \rfloor = 1 + \lfloor \log\left(9.69 \times 10^5\right) \rfloor \\ &= 20, \; Iter_4 = 1 + \lfloor \log\left(\frac{ME_4}{ME_{min}}\right) \rfloor \\ &= 1 + \lfloor \log\left(1.02 \times 10^8\right) \rfloor = 27, \end{aligned}$$

The calculated $\mu_i^\star$ values of 16 variables and $Iter_s$ of the subcomponents are given in Fig.1a.

Example 2 For $k = 4$ and $n = 16$, consider a non-separable function which is expressed as follows:

$$\begin{aligned} f(\mathbf{X}) = {} & 10^{4}.f_{rastrigin}(x_1, x_2, x_3, x_4) \\ & + 10^{4}.f_{rastrigin}(x_5, x_6, x_7, x_8) \\ & + 10^{4}.f_{rastrigin}(x_9, x_{10}, x_{11}, x_{12}) \\ & + 10^{4}.f_{rastrigin}(x_{13}, x_{14}, x_{15}, x_{16}) \end{aligned}$$

This non-separable function has four subcomponents with the equal effects. First, $\mu_i^\star$ values of 16 variables are computed by Morris method. Table 2 shows $\mu_i^\star$ values. Then,

Table 1 $\mu_i^\star$ values of 16 variables

Variables	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
$\mu_i^\star$ values	2.32×10^{-3}	2.28×10^{-3}	2.31×10^{-3}	$2.32e\times10^{-3}$	2.31×10^{-1}	2.31×10^{-1}	2.25×10^{-1}	2.59×10^{-1}
Variables	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}
$\mu_i^\star$ values	2.44×10^{3}	2.27×10^{3}	2.33×10^{3}	1.91×10^{3}	2.33×10^{5}	2.45×10^{5}	2.24×10^{5}	2.44×10^{5}

ME_s values of four subcomponents are computed and we have:

$$\begin{aligned} ME_1 &= \frac{\sum_{i=1}^{4}\mu_i^\star}{\sum_{i=1}^{16}\mu_i} = 2.41\times10^{-1},\ ME_2 \\ &= \frac{\sum_{i=5}^{8}\mu_i^\star}{\sum_{i=1}^{16}\mu_i} = 2.47\times10^{-1}, \\ ME_3 &= \frac{\sum_{i=9}^{12}\mu_i^\star}{\sum_{i=1}^{16}\mu_i} = 2.64\times10^{-1},\ ME_4 \\ &= \frac{\sum_{i=13}^{16}\mu_i^\star}{\sum_{i=1}^{16}\mu_i} = 2.47\times10^{-1}, \end{aligned}$$

Thus,

$$\begin{aligned} Iter_1 &= 1+\lfloor\log\left(\frac{ME_1}{ME_{min}}\right)\rfloor = 1+\lfloor\log(1)\rfloor = 1,\ Iter_2 \\ &= 1+\lfloor\log\left(\frac{ME_2}{ME_{min}}\right)\rfloor = 1+\lfloor\log(1.02)\rfloor = 1, \\ Iter_3 &= 1+\lfloor\log\left(\frac{ME_3}{ME_{min}}\right)\rfloor = 1+\lfloor\log(1.1)\rfloor = 1,\ Iter_4 \\ &= 1+\lfloor\log\left(\frac{ME_4}{ME_{min}}\right)\rfloor = 1+\lfloor\log(1.02)\rfloor = 1, \end{aligned}$$

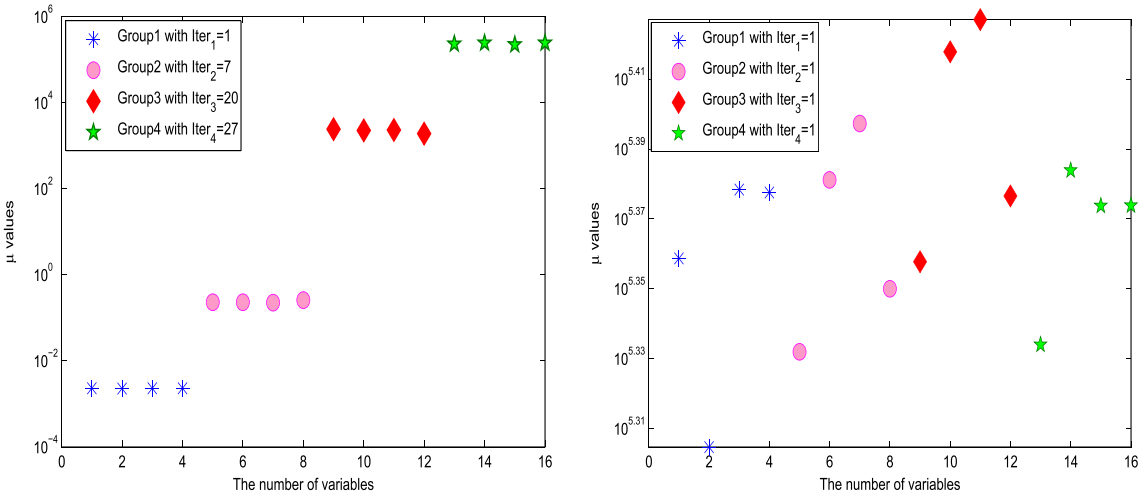
The calculated $\mu_i^\star$ values of 16 variables and $Iter_s$ of the subcomponents are given in Fig.1b.

Example 3 For $k = 4$ and $n = 16$, consider a non-separable function which is expressed as follows:

$$\begin{aligned} f(\mathbf{X}) &= 10^2.f_{rastrigin}(x_1,x_2,x_3,x_4) \\ &\quad +10^{-2}.f_{rastrigin}(x_5,x_6,x_7,x_8) \\ &\quad +10^{2}.f_{rastrigin}(x_9,x_{10},x_{11},x_{12}) \\ &\quad +10^{4}.f_{rastrigin}(x_{13},x_{14},x_{15},x_{16}) \end{aligned}$$

This non-separable function has two subcomponents with the equal effects and two subcomponents with the different effects. First, $\mu_i^\star$ values of 16 variables are computed by Morris method. Table 3 shows $\mu_i^\star$ values. Then, ME_s values of four subcomponents are computed and we have:

$$ME_1 = \frac{\sum_{i=1}^{4}\mu_i^\star}{\sum_{i=1}^{16}\mu_i} = 9.89\times10^{-3},\ ME_2 = \frac{\sum_{i=5}^{8}\mu_i^\star}{\sum_{i=1}^{16}\mu_i} = 1.07\times10^{-6},$$
$$ME_3 = \frac{\sum_{i=9}^{12}\mu_i^\star}{\sum_{i=1}^{16}\mu_i} = 1.06\times10^{-2},\ ME_4 = \frac{\sum_{i=13}^{16}\mu_i^\star}{\sum_{i=1}^{16}\mu_i} = 9.80\times10^{-1},$$



(a) Example 1 (b) Example 2

Fig. 1 $\mu_i^\star$ values of 16 variables and $Iter_s$ values of the four groups

Table 2 $\mu_i^\star$ values of 16 variables

Variables	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
$\mu_i^\star$ values	2.28×10^5	2.02×10^5	2.39×10^5	2.39×10^5	2.15×10^5	2.41×10^5	2.50×10^5	2.24×10^5
Variables	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}
$\mu_i^\star$ values	2.28×10^5	2.62×10^5	2.67×10^5	2.38×10^5	2.16×10^5	2.42×10^5	2.36×10^5	2.37×10^5

Thus,

$$Iter_1 = 1 + \lfloor \log\left(\frac{ME_1}{ME_{min}}\right)\rfloor = 1 + \lfloor \log\left(9.24 \times 10^3\right)\rfloor = 14,$$

$$Iter_2 = 1 + \lfloor \log\left(\frac{ME_2}{ME_{min}}\right)\rfloor = 1 + \lfloor \log\left(1\right)\rfloor = 1,$$

$$Iter_3 = 1 + \lfloor \log\left(\frac{ME_3}{ME_{min}}\right)\rfloor = 1 + \lfloor \log\left(9.91 \times 10^3\right)\rfloor = 14,\ Iter_4 = 1 + \lfloor \log\left(\frac{ME_4}{ME_{min}}\right)\rfloor = 1 + \lfloor \log\left(9.16 \times 10^5\right)\rfloor = 20,$$

The calculated $\mu_i^\star$ values of 16 variables and $Iter_s$ of the subcomponents are given in Fig. 2.

4 Experimental studies and discussions

4.1 Experimental setup

In order to evaluate the performance of the proposed sensitivity analysis-based CC algorithm, we have utilized the same set of modified CEC-2010 benchmark functions used by Omidvar et al. [18] and also a new set of the benchmark functions based on the CEC-2010 benchmark functions. The CEC-2010 benchmark functions were provided by the CEC-2010 Special Session and Competition on LSGO [35]. In this test set, there are five types of functions as follows:

- Separable functions (f_1–f_3)
- Single-group m-non-separable functions (f_4–f_8)
- $\frac{n}{2m}$-group m-non-separable functions (f_9–f_{13})
- $\frac{n}{m}$-group m-non-separable functions (f_{14}–f_{18})
- Non-separable functions (f_{19}–f_{20})

Where n is the dimension of the function and m is the number of the variables in each non-separable subcomponent. The CEC-2010 benchmark functions have some functions (f_4–f_8) with an imbalanced non-separable subcomponent. In the modified CEC-2010 benchmark functions, Omidvar et al. created an imbalance effect by multiplying some coefficients to the subcomponents of the CEC-2010 benchmark functions (f_9–f_{18}). These coefficients are the fixed powers of 10 and we added the imbalanced functions f_4–f_8 from CEC-2010 along with the modified CEC-2010 benchmark functions in our all experiments. Also, we generated a new set of the benchmark functions, i.e., the modified CEC-2010 test functions with normal weights, which is similar to the modified CEC-2010 test functions while a component is multiplied to a normal distribution coefficient to create an imbalance normal effect. In the modified CEC-2010 test functions, the coefficients of subcomponents are the same for all functions while in the modified CEC-2010 test functions with normal weights, the different imbalance effects are generated by the normal coefficients for all functions. The normal coefficient for ith non-separable subcomponent is calculated as following formula proposed in [11]:

$$C_i = 10^{3N(0,1)} \tag{5}$$

The normal coefficients are provided in the Appendix. In addition, some experiments are conducted on the CEC-2013 LSGO benchmark functions [11] (on the imbalance non-separable functions, i.e., f_4–f_{11} and f_{13}–f_{14}) with new challenging 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 (f_1–f_3), partially separable functions with a separable subcomponent (f_4–f_7), partially separable functions with no separable subcomponents (f_8–f_{11}), overlapping functions (f_{12}–f_{14}) and one fully

Table 3 $\mu_i^\star$ values of 16 variables

Variables	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
$\mu_i^\star$ values	2.34×10^3	2.23×10^3	2.22×10^3	2.33×10^3	2.60×10^{-1}	2.34×10^{-1}	2.43×10^{-1}	2.50×10^{-1}
Variables	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}
$\mu_i^\star$ values	2.48×10^3	2.33×10^3	2.31×10^3	2.62×10^3	2.25×10^5	2.34×10^5	2.14×10^5	2.29×10^5

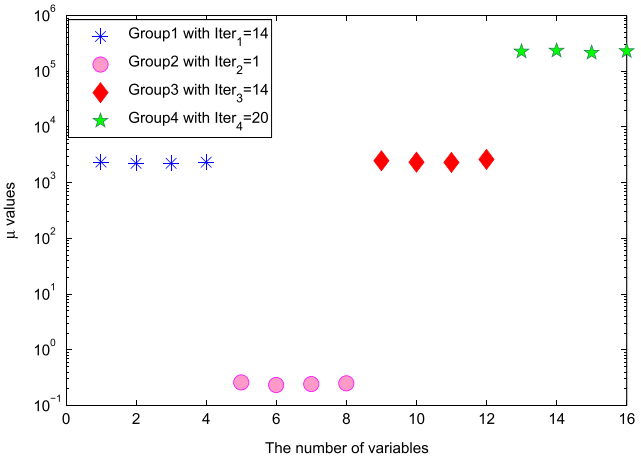


Fig. 2 $\mu_i^\star$ values of 16 variables and $Iter_s$ values of the four groups

non-separable function (f_{15}). For fair comparison among CC algorithms, all compared CC algorithms should be the same optimizer algorithm and also all setting of parameters' optimizer algorithm should be equal. Therefore, we applied a self-adaptive evolution differential with the neighbourhood search strategy (SaNSDE) [41] as optimizer algorithm in all CC algorithms. Also, in this study, the maximum number of evaluations (FES) was set to 3×10^6, the population size (NP) was set to 50, and all algorithms were evaluated for 25 independent runs and the results were recorded. All setting of optimizer algorithm and its parameters are similar to [18]. It should be noted that in comparison with all compared algorithms, we run the compared methods by using the source code of the CC algorithm with DG decomposition method (DECC-DG) in http://goanna.cs.rmit.edu.au/~xiaodong/publications.php. In SACC methods, the parameters r and p were set to 20 and 8 in the Morris method as mentioned above.

4.2 Comparison of SACC methods (SACC1, SACC2, and SACC3) with the standard CC

In this section, we present the experimental results for CC algorithms with three proposed types of SACC; and we compare them against the standard round-robin CC algorithm. To make the comparison fair, compared CC algorithms used the same decomposition method to construct subcomponents. To test whether the sensitivity analysis-based budget division technique can improve the performance of CC algorithms, we tested two decomposition methods, namely, DG and ideal grouping for CC algorithms. Ideal grouping method constructs subcomponents manually using the knowledge of benchmark functions. Due to the specific properties of the CEC-2013 LSGO benchmark functions, as mentioned above, DG decomposition method performed poorly on these functions; thus we use only ideal decomposition method to make a fair comparison in all experiments on these test functions. We performed some statistical tests used in [19] for comparing multiple algorithms. First, the Kruskal-Wallis nonparametric one-way ANOVA [32] is applied to find significant differences among algorithms [19]. When this test finds a significant difference, a series of pair-wise Wilcoxon rank-sum [32] tests at a 0.05 significance level are performed with the Holm p-value correction [32] in order to account for the family-wise error rate. As mentioned in [19], only median is considered because the used statistical tests are nonparametric and all statistically similar algorithms are marked in **bold** face which are not outperformed by the other compared algorithms.

4.2.1 Experiment series 1: results with the DG decomposition method

Table 4 shows the results of the CC algorithm with three proposed sensitivity analysis-based budget division methods and the standard round-robin CC algorithm (DECC-DG) along with the DG decomposition method on the modified CEC-2010 test functions and the modified CEC-2010 test functions with normal weights. It can be seen from Table 4a on the modified CEC-2010 test functions, SACC3 is the best performing algorithm and outperforms other algorithms on 9 out of 15 functions while DECC-DG, SACC1, and SACC2 obtain best results for 4, 7, and 4 functions, respectively. Note that according to obtained p-values of Holm correction SACC1 and SACC2 also outperform statistically DECC-DG on 3 (f_5, f_7, and f_9) and 3 (f_5, f_9, and f_{17}) functions, respectively. On f_{11}, SACC3 does not work as well as DECC-DG. One possible reason for that behavior is that the Morris method cannot correctly identify important variables so that the four maximum subcomponents of this function are misplaced. It should be noted that the performance of SACC3 is more sensitive to high accuracy of the Morris method than SACC1 and SACC2; thus they have results better than, or similar to DECC-DG on f_{11}. On f_{18} and f_{16}, an ideal decomposition method should construct 20 non-separable subcomponents, but the DG decomposition method constructs 12 and 24 non-separable subcomponents for f_{18} and f_{16}, respectively. Therefore, one possible reason why SACC3 cannot correctly assign the budget is that the DG decomposition method has the poor accuracy.

It is obvious from Table 4b on the modified CEC-2010 test functions with normal weights that SACC3 as the best algorithm outperforms other algorithms on 9 out of 15 functions. DECC-DG, SACC1, and SACC2 obtain best results for 5, 6, and 8 out of 15 functions, respectively. Note that according to obtained p-values of Holm correction SACC1 and SACC2 also outperform statistically DECC-DG on 3

Table 4 Results of SACC and standard CC algorithm with DG decomposition method

(a) The modified CEC-2010 test functions						(b) The modified CEC-2010 test functions with normal weights					
Function		DECC-DG	SACC1	SACC2	SACC3	Function		DECC-DG	SACC1	SACC2	SACC3
f_4	Median	1.93e+12	2.01e+12	1.82e+12	**9.40e+10**	f_4	Median	3.18e+11	1.71e+11	1.55e+11	**7.96e+10**
	Mean	2.08e+12	1.96e+12	1.99e+12	9.57e+10		Mean	3.27e+11	1.80e+11	1.77e+11	8.32e+10
	Std	6.76e+11	5.18e+11	6.68e+11	3.13e+10		Std	1.42e+11	5.98e+10	6.78e+10	3.17e+10
f_5	Median	1.28e+08	1.11e+08	1.12e+08	**9.72e+07**	f_5	Median	2.31e+04	**2.05e+04**	**2.00e+04**	**1.88e+04**
	Mean	1.36e+08	1.15e+08	1.18e+08	9.94e+07		Mean	2.32e+04	2.08e+04	1.99e+04	1.96e+04
	Std	2.24e+07	1.60e+07	2.39e+07	1.77e+07		Std	3.33e+03	3.08e+03	2.06e+03	2.49e+03
f_6	Median	**1.63e+01**	**1.64e+01**	**1.64e+01**	1.79e+01	f_6	Median	**1.64e+01**	**1.64e+01**	**1.64e+01**	1.71e+01
	Mean	1.63e+01	1.63e+01	1.64e+01	1.79e+01		Mean	1.64e+01	1.64e+01	1.64e+01	1.71e+01
	Std	3.15e−01	3.98e−01	2.88e−01	2.78e−01		Std	2.91e−01	3.23e−01	4.06e−01	3.12e−01
f_7	Median	6.63e+00	**5.69e−01**	3.67e+00	3.59e+04	f_7	Median	5.56e+03	7.56e+02	**1.81e+02**	6.76e+04
	Mean	2.60e+04	6.67e−01	4.61e+00	3.48e+04		Mean	6.58e+03	1.26e+03	7.14e+02	6.97e+04
	Std	5.38e+04	5.05e−01	2.94e+00	8.80e+03		Std	4.31e+03	1.26e+03	9.87e+02	1.41e+04
f_8	Median	1.69e+06	**2.84e+04**	1.48e+06	2.65e+06	f_8	Median	**7.89e+04**	1.08e+05	**7.68e+04**	3.42e+05
	Mean	1.86e+06	5.11e+05	1.57e+06	2.08e+06		Mean	1.64e+05	1.39e+05	7.02e+04	6.14e+05
	Std	1.12e+06	1.32e+06	1.49e+06	1.76e+06		Std	2.46e+05	1.22e+05	5.17e+04	4.65e+05
f_9	Median	3.38e+11	2.25e+11	2.66e+11	**1.41e+10**	f_9	Median	1.01e+11	**6.84e+10**	**7.33e+10**	**4.32e+10**
	Mean	3.45e+11	2.37e+11	2.64e+11	1.51e+10		Mean	1.10e+11	6.85e+10	6.91e+10	5.28e+10
	Std	7.20e+10	4.85e+10	5.98e+10	5.12e+09		Std	6.44e+10	1.83e+10	2.25e+10	3.05e+10
f_{10}	Median	2.79e+07	2.62e+07	2.77e+07	**9.65e+06**	f_{10}	Median	3.03e+05	3.06e+05	3.00e+05	**1.21e+05**
	Mean	2.80e+07	2.68e+07	2.80e+07	9.55e+06		Mean	3.04e+05	3.01e+05	2.99e+05	1.22e+05
	Std	2.18e+06	2.74e+06	1.65e+06	1.60e+06		Std	1.54e+04	1.85e+04	1.82e+04	2.20e+04
f_{11}	Median	**1.06e+01**	**1.06e+01**	**1.04e+01**	1.69e+01	f_{11}	Median	**1.07e+01**	**1.09e+01**	**1.05e+01**	**1.09e+01**
	Mean	1.07e+01	1.06e+01	1.04e+01	1.79e+01		Mean	1.05e+01	1.08e+01	1.05e+01	1.09e+01
	Std	7.67e-01	8.06e-01	7.86e-01	3.06e+00		Std	6.51e-01	7.59e-01	9.81e-01	7.70e-01
f_{12}	Median	4.55e+07	4.40e+07	4.64e+07	**1.91e+06**	f_{12}	Median	1.53e+07	1.40e+07	1.93e+07	**1.79e+02**
	Mean	5.12e+07	4.64e+07	4.79e+07	3.18e+06		Mean	1.60e+07	1.45e+07	2.08e+07	4.02e+02
	Std	3.19e+07	1.03e+07	8.80e+06	3.68e+06		Std	5.01e+06	3.53e+06	8.66e+06	4.57e+02
f_{13}	Median	**1.10e+07**	**1.10e+07**	**1.11e+07**	**1.17e+07**	f_{13}	Median	**1.08e+07**	**1.02e+07**	1.98e+07	2.40e+07
	Mean	1.06e+07	9.91e+06	1.13e+07	1.15e+07		Mean	1.61e+07	1.34e+07	1.92e+07	7.96e+07
	Std	3.14e+06	4.36e+06	4.96e+06	6.62e+06		Std	8.36e+06	7.38e+06	8.93e+06	1.20e+08
f_{14}	Median	6.42e+12	6.08e+12	5.63e+12	**4.03e+12**	f_{14}	Median	2.24e+13	2.06e+13	2.26e+13	**2.00e+12**
	Mean	6.43e+12	6.19e+12	5.92e+12	4.50e+12		Mean	2.27e+13	2.17e+13	2.41e+13	2.15e+12
	Std	1.41e+12	1.56e+12	1.48e+12	1.01e+12		Std	8.58e+12	6.02e+12	6.48e+12	8.14e+11
f_{15}	Median	1.98e+08	1.98e+08	2.00e+08	**6.79e+07**	f_{15}	Median	5.93e+07	5.89e+07	5.91e+07	**2.02e+07**
	Mean	1.98e+08	1.96e+08	2.01e+08	6.86e+07		Mean	5.90e+07	5.88e+07	5.89e+07	2.14e+07
	Std	8.05e+06	1.22e+07	8.74e+06	1.54e+07		Std	4.22e+06	3.81e+06	4.29e+06	4.88e+06
f_{16}	Median	**1.51e-02**	**1.48e-02**	**1.58e-02**	1.23e+05	f_{16}	Median	1.01e+00	8.98e-01	**6.86e-01**	2.59e+02
	Mean	2.15e-02	2.45e-02	2.65e-02	2.66e+05		Mean	1.13e+00	8.78e-01	7.29e-01	5.15e+03
	Std	2.22e-02	2.73e-02	3.02e-02	3.28e+05		Std	4.19e-01	2.37e-01	2.08e-01	1.85e+04
f_{17}	Median	3.29e+09	2.66e+09	3.31e+09	**1.39e+09**	f_{17}	Median	1.48e+10	1.23e+10	2.00e+10	**3.48e+02**
	Mean	3.80e+09	2.77e+09	3.25e+09	1.51e+09		Mean	1.54e+10	1.46e+10	2.15e+10	7.09e+02
	Std	1.83e+09	6.63e+08	5.18e+08	6.35e+08		Std	7.32e+09	5.72e+09	7.27e+09	1.29e+03
f_{18}	Median	3.24e+08	**1.30e+08**	3.16e+08	1.03e+10	f_{18}	Median	**4.52e+10**	**4.25e+10**	**4.74e+10**	9.61e+10
	Mean	4.32e+08	1.40e+08	4.46e+08	1.15e+10		Mean	7.84e+10	5.78e+10	4.85e+10	2.63e+11
	Std	2.49e+08	7.19e+07	3.22e+08	6.60e+09		Std	1.05e+11	4.47e+10	1.55e+10	3.30e+11

The highlighted entries are significantly better

Table 5 Results of SACC and standard CC algorithm with the ideal decomposition method

(a) The modified CEC-2010 test functions

Function		DECC-I	SACC1	SACC2	SACC3
f_4	Median	**4.93e+11**	5.78e+11	**5.07e+11**	**3.44e+11**
	Mean	5.33e+11	5.73e+11	5.40e+11	4.01e+11
	Std	2.59e+11	2.00e+11	2.84e+11	2.26e+11
f_5	Median	**1.12e+08**	1.20e+08	1.22e+08	**1.06e+08**
	Mean	1.14e+08	1.21e+08	1.20e+08	1.07e+08
	Std	1.87e+07	1.60e+07	1.82e+07	1.80e+07
f_6	Median	**1.64e+01**	**1.63e+01**	**1.64e+01**	1.79e+01
	Mean	1.63e+01	1.63e+01	1.64e+01	1.78e+01
	Std	3.63e-01	2.41e-01	3.31e-01	3.57e-01
f_7	Median	**5.56e+02**	**4.73e+02**	**3.47e+02**	7.87e+05
	Mean	1.15e+03	1.04e+03	5.64e+02	7.85e+05
	Std	1.67e+03	1.63e+03	8.52e+02	4.70e+04
f_8	Median	**3.69e+02**	**7.48e+02**	**5.11e+02**	9.27e+05
	Mean	3.20e+05	2.06e+03	1.61e+05	1.36e+06
	Std	1.10e+06	2.97e+03	7.97e+05	1.22e+06
f_9	Median	1.79e+11	1.75e+11	1.78e+11	**1.44e+10**
	Mean	1.89e+11	1.77e+11	1.80e+11	1.65e+10
	Std	3.70e+10	2.78e+10	4.09e+10	5.79e+09
f_{10}	Median	2.41e+07	2.43e+07	2.35e+07	**8.33e+06**
	Mean	2.39e+07	2.38e+07	2.38e+07	8.61e+06
	Std	1.64e+06	2.11e+06	2.26e+06	1.44e+06
f_{11}	Median	**1.05e+01**	**9.83e+00**	**1.02e+01**	8.97e+02
	Mean	1.06e+01	1.01e+01	1.02e+01	1.95e+04
	Std	1.04e+00	9.09e-01	1.15e+00	4.01e+04
f_{12}	Median	2.87e+06	3.22e+06	2.92e+06	**4.77e+04**
	Mean	2.91e+06	3.34e+06	3.00e+06	4.80e+04
	Std	1.04e+06	1.10e+06	8.21e+05	1.33e+04
f_{13}	Median	1.77e+06	**1.04e+06**	1.72e+06	**7.17e+05**
	Mean	1.73e+06	9.48e+05	1.56e+06	8.33e+05
	Std	3.99e+05	3.06e+05	5.23e+05	2.00e+05
f_{14}	Median	5.34e+12	5.01e+12	4.96e+12	**3.12e+11**
	Mean	5.44e+12	5.08e+12	5.31e+12	1.40e+12
	Std	1.10e+12	8.10e+11	1.16e+12	5.44e+12
f_{15}	Median	1.72e+08	1.73e+08	1.76e+08	**6.84e+07**
	Mean	1.71e+08	1.71e+08	1.74e+08	6.56e+07
	Std	1.21e+07	1.20e+07	1.12e+07	1.82e+07
f_{16}	Median	**8.79e-09**	1.32e-08	**9.35e-09**	5.89e+05
	Mean	8.61e-09	1.35e-08	9.34e-09	4.66e+05
	Std	1.92e-09	2.90e-09	1.60e-09	3.75e+05
f_{17}	Median	4.56e+08	4.92e+08	4.51e+08	**1.18e+02**
	Mean	4.70e+08	4.85e+08	4.66e+08	2.45e+02
	Std	8.80e+07	8.54e+07	7.63e+07	6.45e+02
f_{18}	Median	2.74e+07	1.85e+07	2.45e+07	**1.08e+07**
	Mean	3.29e+07	2.31e+07	2.79e+07	1.42e+07
	Std	1.56e+07	1.14e+07	1.11e+07	1.21e+07

(b) The modified CEC-2010 test functions with normal weights

Function		DECC-I	SACC1	SACC2	SACC3
f_4	Median	4.80e+10	**3.61e+10**	4.57e+10	**3.43e+10**
	Mean	5.12e+10	4.39e+10	4.83e+10	3.62e+10
	Std	2.18e+10	2.20e+10	1.46e+10	1.45e+10
f_5	Median	2.38e+04	2.07e+04	2.16e+04	**1.82e+04**
	Mean	2.29e+04	2.11e+04	2.14e+04	1.88e+04
	Std	2.92e+03	2.76e+03	2.60e+03	2.35e+03
f_6	Median	**1.62e+01**	**1.64e+01**	**1.64e+01**	1.75e+01
	Mean	1.62e+01	1.64e+01	1.63e+01	1.74e+01
	Std	3.34e-01	4.56e-01	4.06e-01	3.52e-01
f_7	Median	**3.75e+00**	5.75e+02	8.17e+02	7.22e+04
	Mean	2.07e+01	8.01e+02	1.09e+03	7.20e+04
	Std	4.10e+01	7.46e+02	1.02e+03	1.07e+04
f_8	Median	**2.01e+01**	1.40e+03	2.83e+02	7.42e+05
	Mean	6.48e+02	2.43e+03	1.31e+03	7.41e+05
	Std	1.67e+03	2.43e+03	1.81e+03	4.02e+04
f_9	Median	7.87e+10	3.82e+10	4.02e+10	**3.71e+09**
	Mean	8.15e+10	3.70e+10	3.93e+10	3.91e+09
	Std	2.46e+10	8.87e+09	9.20e+09	1.18e+09
f_{10}	Median	3.13e+05	2.90e+05	2.86e+05	**1.15e+05**
	Mean	3.12e+05	2.87e+05	2.83e+05	1.11e+05
	Std	1.94e+04	1.56e+04	1.93e+04	1.95e+04
f_{11}	Median	**1.04e+01**	**1.00e+01**	**1.08e+01**	**1.04e+01**
	Mean	1.03e+01	1.02e+01	1.06e+01	1.07e+01
	Std	1.02e+00	8.66e-01	8.17e-01	9.71e-01
f_{12}	Median	1.73e+07	3.91e+05	4.20e+05	**5.21e+04**
	Mean	1.83e+07	3.98e+05	4.54e+05	5.18e+04
	Std	6.04e+06	6.80e+04	9.99e+04	1.48e+04
f_{13}	Median	6.39e+06	4.65e+05	2.61e+06	**1.64e+05**
	Mean	8.94e+06	5.23e+05	2.77e+06	9.85e+05
	Std	4.90e+06	3.26e+05	2.79e+06	3.03e+06
f_{14}	Median	9.37e+13	2.10e+13	2.18e+13	**2.21e+12**
	Mean	9.43e+13	2.21e+13	2.63e+13	2.31e+12
	Std	2.92e+13	7.96e+12	1.10e+13	8.31e+11
f_{15}	Median	6.43e+07	5.63e+07	5.63e+07	**1.99e+07**
	Mean	6.36e+07	5.61e+07	5.59e+07	2.02e+07
	Std	3.34e+06	3.79e+06	3.22e+06	4.95e+06
f_{16}	Median	**1.35e-08**	1.45e-08	**1.13e-08**	9.32e+01
	Mean	1.48e-08	1.39e-08	1.13e-08	1.40e+04
	Std	2.52e-09	3.63e-09	1.86e-09	3.90e+04
f_{17}	Median	1.39e+11	3.67e+09	5.74e+09	**7.05e+01**
	Mean	1.44e+11	4.23e+09	6.18e+09	1.53e+03
	Std	3.36e+10	1.81e+09	2.04e+09	6.39e+03
f_{18}	Median	1.22e+10	6.28e+09	9.18e+09	**2.52e+09**
	Mean	1.45e+10	8.77e+09	1.11e+10	3.64e+09
	Std	5.37e+09	4.65e+09	5.84e+09	4.06e+09

The highlighted entries are significantly better

(f_4, f_7, and f_{12}) and 2 (f_4 and f_{17}) functions, respectively. The DG decomposition method constructs 33 non-separable subcomponents on both f_{13} and f_{18}; while they have 10 and 20 non-separable subcomponents, respectively. Therefore, the poor performance of SACC3 may be due to low accuracy of DG decomposition on these functions. As it can be seen from Table 4, SACC3 was effective for most test functions with 10 or 20 imbalanced subcomponents; because it has the ability of handling budget assignment.

4.2.2 Experiment series 2: results with the ideal decomposition method

The results of CC algorithm with three proposed types of SACC and the standard round-robin CC algorithm (DECC-I) which use the ideal decomposition method are summarized in Table 5. It can be seen from Table 5a that on the modified CEC-2010 test functions, SACC3 is the best performing algorithm and outperforms other algorithms on 10 out of 15 functions while DECC-I, SACC1, and SACC2 obtain best results for 7, 5, and 6 functions, respectively. From Table 5b, it is obvious on the modified CEC-2010 test functions with normal weights that SACC3 as the best algorithm outperforms other algorithms on 11 out of 15 functions while DECC-I, SACC1, and SACC2 obtain best results for 5, 3, and 3 out of 15 functions, respectively. It is also notable that while SACC1 and SACC2 cannot perform as best algorithm in comparison with all compared algorithms on most functions, SACC1 and SACC2 outperform statistically DECC-I based on obtained p-values of Holm correction on 8 (f_9–f_{10}, f_{12}–f_{15}, and f_{17}–f_{18}) functions.

DECC-I uses all the fitness evaluations for optimization while SACC1, SACC2, and SACC3 methods with ideal grouping use a number of fitness evaluations ($i.e., = r(k + 1) = 20 * 1001 = 20020$) to identify effect of variables on the global fitness. In addition, SACC1 and SACC2 change the assigned budget for only the maximum subcomponent. With the possible reasons, DECC-I can obtain the similar results compared to two types of SACC, SACC1 and SACC2, on the modified CEC-2010 test functions. From the result of the DG decomposition method, the results of SACC3 is worse than the standard round-robin CC algorithm on some functions, f_{13} and f_{18}, in the modified CEC-2010 test functions with normal weights. In the presence of the ideal decomposition method, SACC3 outperforms the standard round-robin CC algorithm on these functions. The same behavior of the CCBC algorithms was shown in [18, 22] that the performance of the CCBC algorithms increases with the ideal decomposition method. Although, in the presence of the ideal decomposition method, the performance of SACC3 still decreases on two functions, f_{11} and f_{16}.

From Table 5, it is clear that SACC3 cannot perform well and may even perform worse than DECC-I on f_4–f_8 function in the modified CEC-2010 test functions and the modified CEC-2010 test functions with normal weights. The possible reason for this behavior is that these functions have one non-separable imbalanced subcomponent; thus using a number of fitness evaluations for the budget assignment might deteriorate the performance of DECC-I. As a general consideration of all results, that performance of SACC3 is either significantly better than or comparable to the standard CC algorithm. The main reason why SACC3 performs better is that it properly considers the computational budget among subcomponents. Table 6 shows the results of CC algorithm with three proposed types of SACC and the standard round-robin CC algorithm (DECC-I) by

Table 6 Results of SACC and standard CC algorithm with the ideal decomposition method for the CEC-2013 LSGO benchmark functions

Function		DECC-I	SACC1	SACC2	SACC3
f_4	Median	4.63e+10	2.18e+10	2.51e+10	**1.60e+10**
	Mean	4.97e+10	2.55e+10	2.59e+10	1.81e+10
	Std	1.97e+10	1.07e+10	1.20e+10	8.53e+09
f_5	Median	4.98e+06	4.53e+06	4.45e+06	**2.35e+06**
	Mean	4.96e+06	4.49e+06	4.45e+06	2.33e+06
	Std	3.63e+05	3.21e+05	4.66e+05	5.02e+05
f_6	Median	**1.38e+01**	**1.38e+01**	**1.42e+01**	8.15e+04
	Mean	1.20e+04	1.26e+04	1.32e+04	7.92e+04
	Std	2.51e+04	2.64e+04	2.41e+04	3.62e+04
f_7	Median	6.67e+07	5.53e+07	4.60e+07	**1.51e+06**
	Mean	6.33e+07	5.67e+07	4.93e+07	1.76e+07
	Std	2.36e+07	2.04e+07	2.67e+07	6.23e+07
f_8	Median	5.16e+15	2.80e+15	3.12e+15	**6.90e+12**
	Mean	4.86e+15	2.74e+15	3.06e+15	1.01e+13
	Std	1.85e+15	7.83e+14	9.39e+14	8.16e+12
f_9	Median	4.97e+08	4.34e+08	4.32e+08	**1.43e+08**
	Mean	4.97e+08	4.38e+08	4.23e+08	1.67e+08
	Std	3.53e+07	2.48e+07	4.22e+07	5.93e+07
f_{10}	Median	**5.28e+00**	**6.25e-01**	**3.89e+00**	1.05e+02
	Mean	1.64e+01	5.29e+00	8.31e+00	1.09e+02
	Std	1.96e+01	8.27e+00	1.33e+01	1.70e+01
f_{11}	Median	1.78e+09	1.56e+09	1.64e+09	**1.11e+08**
	Mean	3.27e+09	1.76e+09	2.58e+09	1.70e+10
	Std	5.11e+09	8.64e+08	2.96e+09	3.86e+10
f_{13}	Median	9.03e+09	7.65e+09	5.47e+09	**1.17e+09**
	Mean	8.96e+09	7.68e+09	5.89e+09	3.31e+09
	Std	2.30e+09	2.83e+09	2.21e+09	3.71e+09
f_{14}	Median	8.55e+10	7.52e+10	8.29e+10	**6.08e+08**
	Mean	8.77e+10	7.76e+10	8.48e+10	2.81e+10
	Std	2.53e+10	1.87e+10	2.25e+10	8.32e+10

The highlighted entries are significantly better

using the ideal decomposition method on the CEC-2013 LSGO benchmark functions. From the results of Table 6, SACC3 is the best performing algorithm and outperforms other algorithms on 8 out of 10 functions while DECC-I, SACC1, and SACC2 obtain best results for 2 functions. It is also notable that while SACC1 and SACC2 perform as the best algorithm, similar to DECC-I, but SACC1 and SACC2 outperform statistically DECC-I based on obtained p-values of Holm correction on 6 (f_4–f_5, f_8–f_9, f_{11}, and f_{13}) and 5 (f_4–f_5, f_8–f_9, and f_{13}) functions, respectively. SACC1 and SACC2 methods have the same performance in comparison with DECC-I on 2 (f_6 and f_{10}) functions. SACC3 is very sensitive (more than SACC1 and SACC2) to Morris method which is used to determine the effect of the variables; because in SACC3, the Morris method should sort subcomponents according to their effects. The possible reason of why SACC3 performs worse than SACC1, SACC2, and DECC-I on (f_6 and f_{10}) is that Morris method cannot correctly identify subcomponents' effects in such functions.

From the study of Tables 4, 5b, and 6, it is clear that CC algorithms with the sensitivity analysis-based budget assignment method gained much better results than the standard round-robin CC algorithm. Also, results confirmed the advantages of sensitivity analysis budget assignment methods when there is an imbalance feature among the non-separable subcomponents in LSGO problems to the global fitness. Important observation about the performance of SACC3 is that on f_4–f_8, functions with one imbalanced subcomponent, it is less significant, but on f_9–f_{13} and f_{14}–f_{18}, functions with 10 or 20 imbalanced subcomponents in the modified CEC-2010 test functions and the modified CEC-2010 test functions with normal weights, and imbalance functions with 7 and 20 imbalanced subcomponents f_4–f_{11} and f_{13}–f_{14} in the CEC-2013 LSGO benchmark functions, it has been concluded that it is either

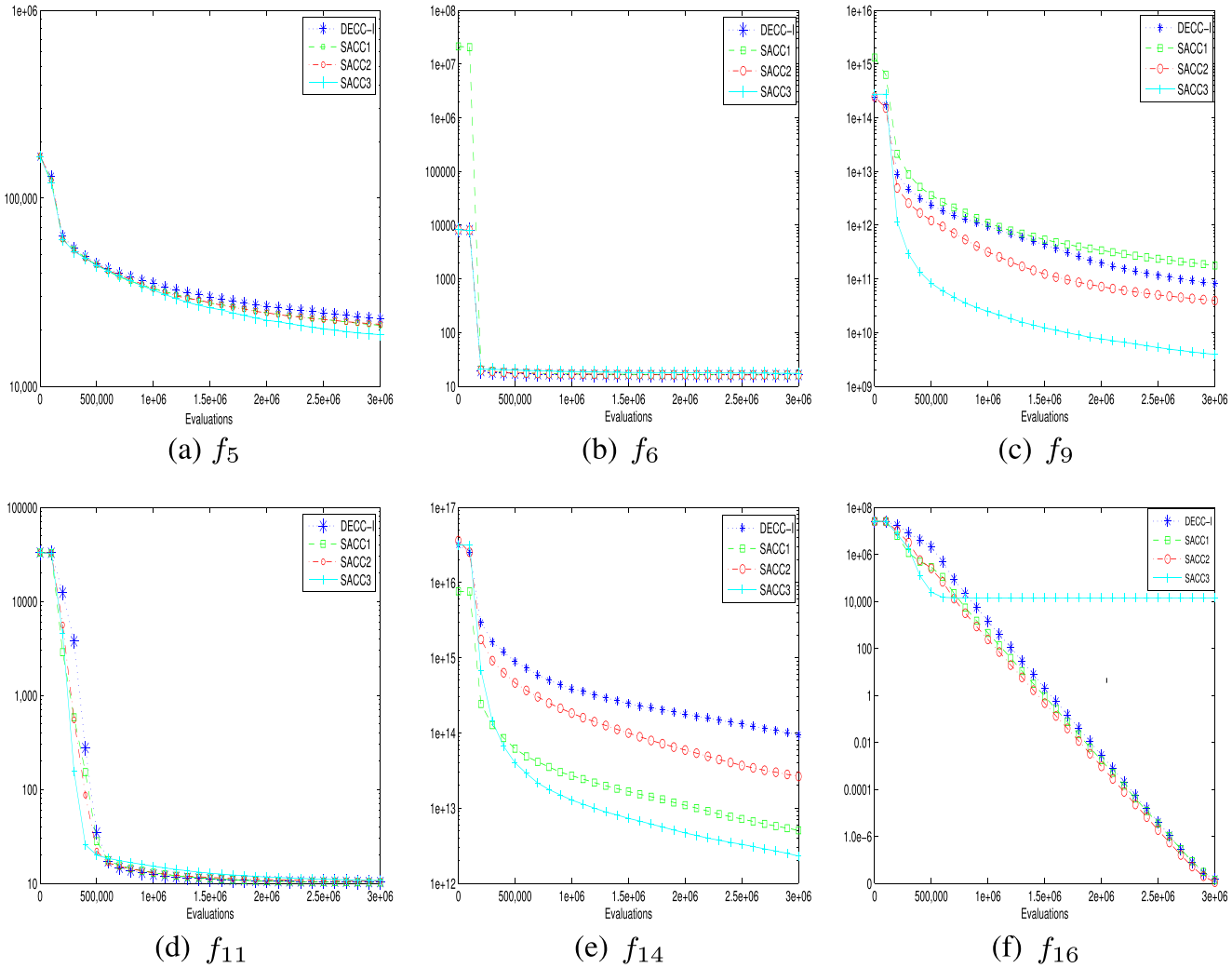


Fig. 3 Convergence plots of f_5, f_6, f_9, f_{11}, f_{14}, and f_{16} of the modified CEC-2010 test functions with normal weights. The results were averaged over 25 runs. The vertical axis is the function value and the horizontal axis is the number of function evaluations

significantly better than or comparable to the standard round-robin CC algorithm for the most of these functions. Therefore, it can be found that it is better to apply the sensitivity analysis-based budget division method for CC algorithms when the number of the imbalanced subcomponents increases.

To gain a better understanding of the behavior of algorithms, we plot the convergence graph for six sample problems (two functions from three classes of functions) on the modified CEC-2010 benchmark functions and the modified CEC-2010 test functions with normal weights in Figs. 3 and 4, respectively. 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 twenty imbalanced subcomponents) are shown in Fig. 5. In addition, to gain a better understanding of how SACC3 can assign $Iter_s$ to subcomponents, the assigned iteration of the subcomponents and their corresponding weights are plotted in Figs. 6, 7, and 8 in the modified CEC-2010 test functions with normal weights.

4.3 Comparison of SACC3 with the contribution based methods (CBCC1 and CBCC2)

As mentioned earlier, we have demonstrated that our proposed budget division methods can improve the performance of CC algorithms. In this section, we selected SACC3 from three proposed sensitivity analysis-based budget assignment methods to compare with CBCC1 and CBCC2 methods proposed by Omidvar et al. [18, 22]. We performed a series of pair-wise Wilcoxon rank-sum [32] tests at a 0.05 significance level with Holm p-value

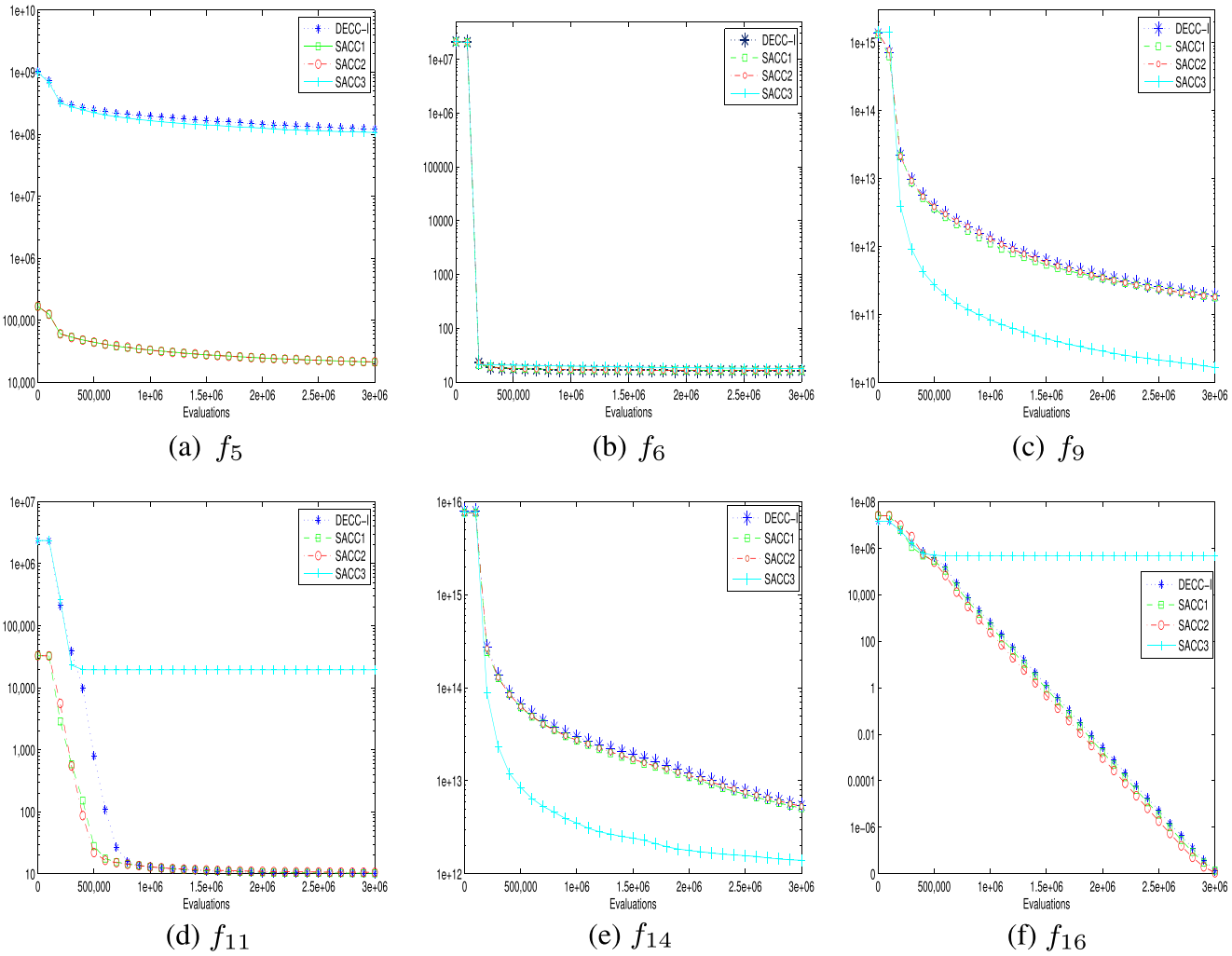


Fig. 4 Convergence plots of f_5, f_6, f_9, f_{11}, f_{14}, and f_{16} of the modified CEC-2010 test functions. The results were averaged over 25 runs. The vertical axis is the function value and the horizontal axis is the number of function evaluations

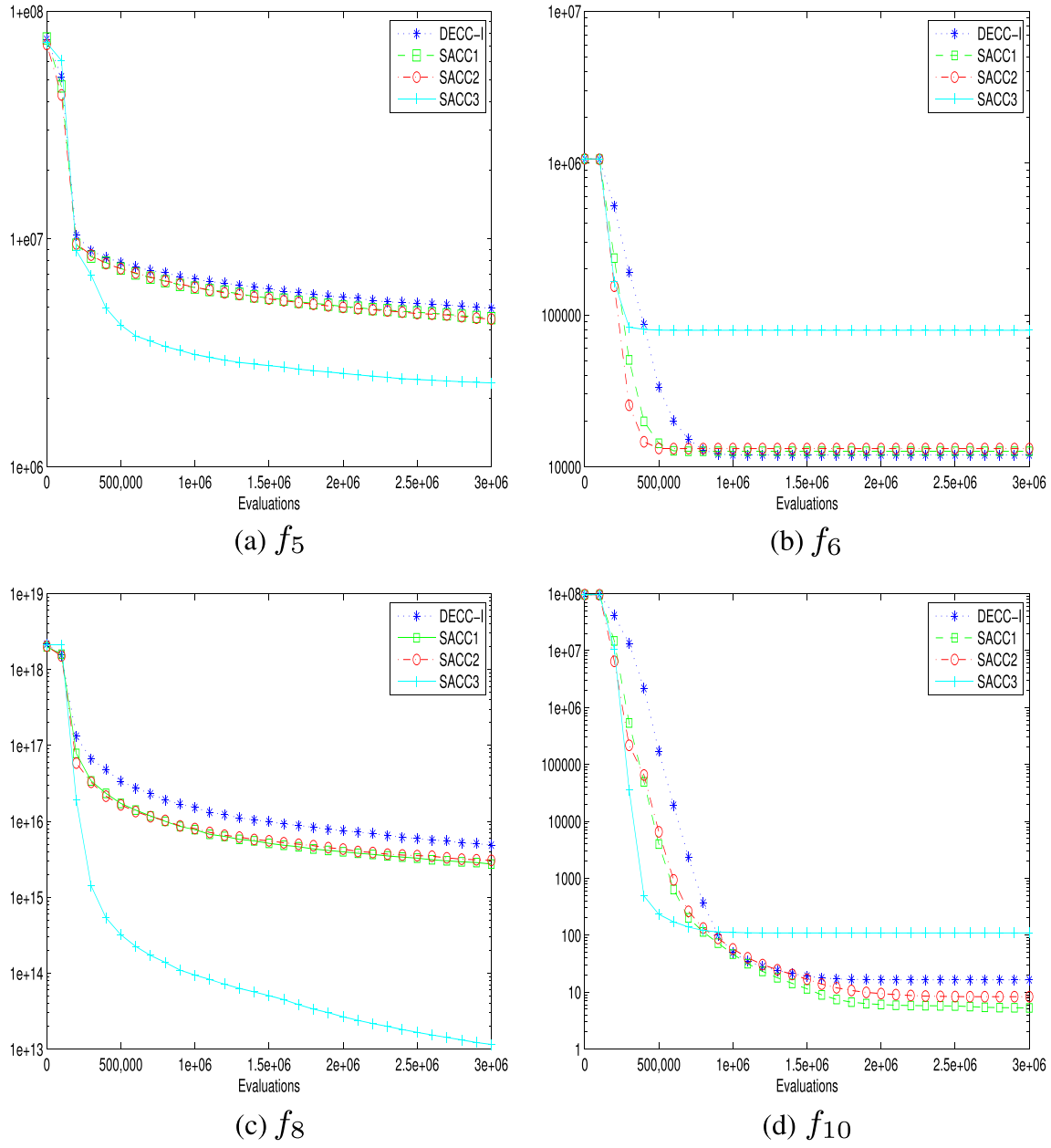


Fig. 5 Convergence plots of f_5, f_6, f_8, and f_{10} of the CEC-2013 LSGO benchmark functions. The results were averaged over 25 runs. The vertical axis is the function value and the horizontal axis is the number of function evaluations

correction to compare algorithms as mentioned above. The results that are statistically best are marked in **bold** face.

4.3.1 Experiment series 1: comparison of methods along with the DG decomposition method

The median, mean and standard deviation of the obtained results by SACC3, CBCC1, and CBCC2 along with the DG decomposition method on the modified CEC-2010 test functions and the modified CEC-2010 test functions with normal weights are summarized in Table 7. On the modified CEC-2010 test functions with normal weights, Table 7a indicates that SACC3 is the best performing algorithm and outperforms other algorithms on 8 out of 15 functions while CCBC1 and CCBC2 obtain best results for 6 and 5 out of 15 functions, respectively. From Table 7b, on the modified CEC-2010 test functions, SACC3 as the best algorithm outperforms other algorithms on 9 out of 15 functions while CCBC1 and CCBC2 obtain best results for 4 and 7 out of 15 functions, respectively. The possible reason for poor behavior of SACC3 on f_{18} and f_{16} is because of a large number of subcomponents which are constructed by DG

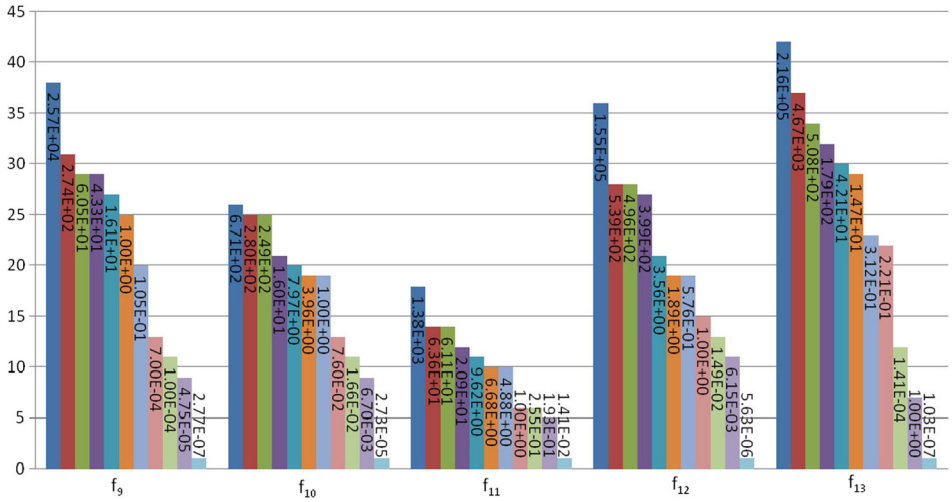


Fig. 6 The assigned iteration for f_9–f_{13} in the modified CEC-2010 test functions. The coefficients of subcomponents are included in the columns of the plot

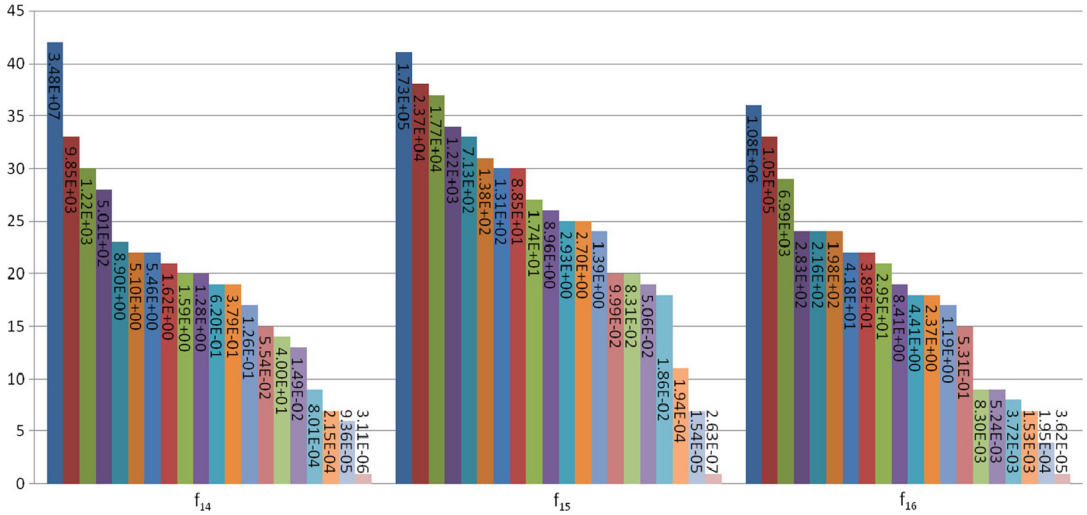


Fig. 7 The assigned iteration for f_{14}–f_{16} in the modified CEC-2010 test functions. The coefficients of subcomponents are included in the columns of the plot

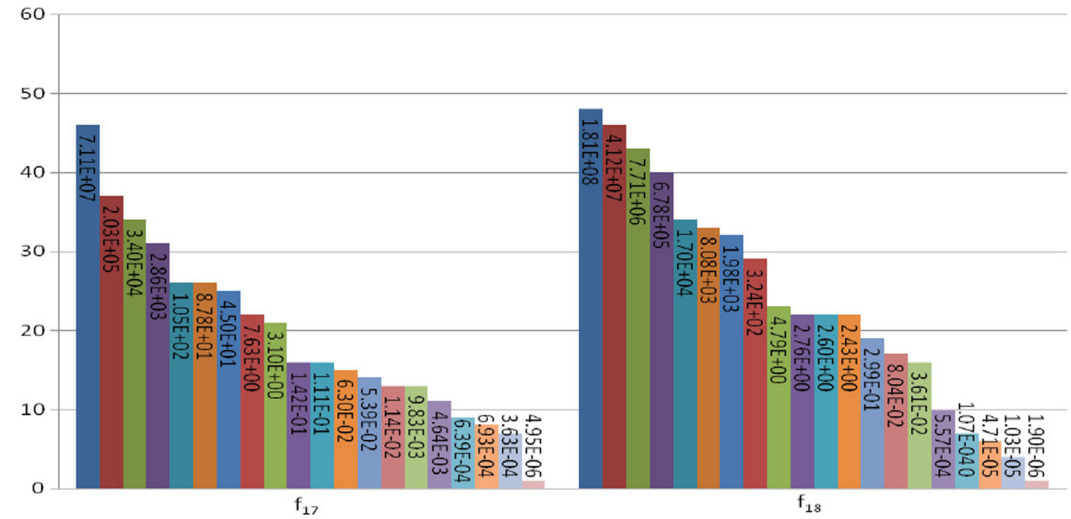


Fig. 8 The assigned iteration for f_{17}–f_{18} in the modified CEC-2010 test functions. The coefficients of subcomponents are included in the columns of the plot

Table 7 Results of SACC3 and contribution based CC algorithm with the DG decomposition method

(a) The modified CEC-2010 test functions with normal weights					The modified CEC-2010 test functions				
Function		CBCC1	CBCC2	SACC3	Function		CBCC1	CBCC2	SACC3
f_4	Median	2.72e+11	2.73e+11	**7.96e+10**	f_4	Median	2.06e+12	1.89e+12	**9.40e+10**
	Mean	2.92e+11	3.01e+11	8.32e+10		Mean	2.21e+12	1.95e+12	9.57e+10
	Std	1.07e+11	1.72e+11	3.17e+10		Std	5.24e+11	6.22e+11	3.13e+10
f_5	Median	**2.06e+04**	**2.02e+04**	**1.88e+04**	f_5	Median	1.21e+08	1.16e+08	**9.72e+07**
	Mean	2.08e+04	2.05e+04	1.96e+04		Mean	1.20e+08	1.14e+08	9.94e+07
	Std	1.99e+03	3.54e+03	2.49e+03		Std	2.03e+07	1.94e+07	1.77e+07
f_6	Median	**1.63e+01**	**1.64e+01**	1.71e+01	f_6	Median	**1.65e+01**	**1.65e+01**	1.79e+01
	Mean	1.63e+01	1.63e+01	1.71e+01		Mean	1.65e+01	1.64e+01	1.79e+01
	Std	3.69e-01	3.79e-01	3.12e-01		Std	3.69e-01	3.00e-01	2.78e-01
f_7	Median	**1.59e+02**	5.46e+02	6.76e+04	f_7	Median	5.24e+00	**7.05e-01**	3.59e+04
	Mean	4.55e+02	1.22e+03	6.97e+04		Mean	8.54e+03	1.76e+03	3.48e+04
	Std	7.89e+02	1.42e+03	1.41e+04		Std	3.01e+04	8.82e+03	8.80e+03
f_8	Median	7.36e+04	**1.15e+04**	3.42e+05	f_8	Median	1.39e+06	**2.83e+04**	2.65e+06
	Mean	1.04e+05	3.06e+04	6.14e+05		Mean	7.17e+06	1.91e+05	2.08e+06
	Std	1.63e+05	4.16e+04	4.65e+05		Std	2.09e+07	7.93e+05	1.76e+06
f_9	Median	1.14e+11	1.28e+11	**4.32e+10**	f_9	Median	2.66e+11	2.39e+11	**1.41e+10**
	Mean	1.22e+11	1.31e+11	5.28e+10		Mean	2.75e+11	2.56e+11	1.51e+10
	Std	5.03e+10	5.18e+10	3.05e+10		Std	4.42e+10	6.29e+10	5.12e+09
f_{10}	Median	3.08e+05	3.00e+05	**1.21e+05**	f_{10}	Median	2.69e+07	2.78e+07	**9.65e+06**
	Mean	3.03e+05	3.00e+05	1.22e+05		Mean	2.78e+07	2.80e+07	9.55e+06
	Std	1.78e+04	1.93e+04	2.20e+04		Std	2.22e+06	1.88e+06	1.60e+06
f_{11}	Median	**1.03e+01**	1.08e+01	1.09e+01	f_{11}	Median	**1.06e+01**	**1.01e+01**	1.69e+01
	Mean	1.03e+01	1.10e+01	1.09e+01		Mean	1.41e+01	1.02e+01	1.79e+01
	Std	6.92e-01	9.07e-01	7.70e-01		Std	1.74e+01	9.10e-01	3.06e+00
f_{12}	Median	1.58e+07	1.33e+07	**1.79e+02**	f_{12}	Median	4.45e+07	3.91e+07	**1.91e+06**
	Mean	1.74e+07	1.44e+07	4.02e+02		Mean	5.04e+07	4.63e+07	3.18e+06
	Std	4.23e+06	5.86e+06	4.57e+02		Std	2.75e+07	2.13e+07	3.68e+06
f_{13}	Median	1.06e+07	**1.02e+07**	2.40e+07	f_{13}	Median	**1.06e+07**	**1.09e+07**	**1.17e+07**
	Mean	1.61e+07	1.42e+07	7.96e+07		Mean	1.02e+07	1.08e+07	1.15e+07
	Std	9.02e+06	7.19e+06	1.20e+08		Std	4.18e+06	5.03e+06	6.62e+06
f_{14}	Median	2.54e+13	2.27e+13	**2.00e+12**	f_{14}	Median	6.10e+12	6.06e+12	**4.03e+12**
	Mean	2.45e+13	2.14e+13	2.15e+12		Mean	6.35e+12	6.13e+12	4.50e+12
	Std	6.46e+12	7.76e+12	8.14e+11		Std	1.39e+12	1.42e+12	1.01e+12
f_{15}	Median	5.99e+07	6.02e+07	**2.02e+07**	f_{15}	Median	1.98e+08	2.00e+08	**6.79e+07**
	Mean	5.97e+07	5.91e+07	2.14e+07		Mean	1.96e+08	2.01e+08	6.86e+07
	Std	3.88e+06	3.67e+06	4.88e+06		Std	1.16e+07	8.82e+06	1.54e+07
f_{16}	Median	**5.92e-01**	8.74e-01	2.59e+02	f_{16}	Median	**1.48e-02**	**1.48e-02**	1.23e+05
	Mean	6.04e-01	1.64e+01	5.15e+03		Mean	2.11e-02	1.83e-02	2.66e+05
	Std	1.30e-01	7.79e+01	1.85e+04		Std	2.59e-02	1.83e-02	3.28e+05
f_{17}	Median	2.21e+10	1.54e+10	**3.48e+02**	f_{17}	Median	2.93e+09	3.03e+09	**1.39e+09**
	Mean	2.34e+10	1.66e+10	7.09e+02		Mean	4.20e+09	3.94e+09	1.51e+09
	Std	7.40e+09	6.89e+09	1.29e+03		Std	2.00e+09	1.57e+09	6.35e+08
f_{18}	Median	**4.96e+10**	**4.34e+10**	9.61e+10	f_{18}	Median	4.72e+08	**1.11e+08**	1.03e+10
	Mean	9.64e+10	1.06e+11	2.63e+11		Mean	5.36e+08	1.51e+08	1.15e+10
	Std	1.70e+11	1.93e+11	3.30e+11		Std	3.85e+08	9.85e+07	6.60e+09

The highlighted entries are significantly better

Table 8 Results of SACC3 and contribution based CC algorithm with the ideal decomposition method

(a) The modified CEC-2010 test functions with normal weights					The modified CEC-2010 test functions				
Function		CBCC1	CBCC2	SACC3	Function		CBCC1	CBCC2	SACC3
f_4	Median	**4.61e+10**	**3.96e+10**	**3.43e+10**	f_4	Median	5.70e+11	6.39e+11	**3.44e+11**
	Mean	4.76e+10	4.74e+10	3.62e+10		Mean	5.77e+11	6.49e+11	4.01e+11
	Std	2.08e+10	2.37e+10	1.45e+10		Std	2.51e+11	3.03e+11	2.26e+11
f_5	Median	**2.07e+04**	**1.89e+04**	**1.82e+04**	f_5	Median	**1.14e+08**	**1.18e+08**	**1.06e+08**
	Mean	2.02e+04	1.96e+04	1.88e+04		Mean	1.16e+08	1.17e+08	1.07e+08
	Std	2.57e+03	2.69e+03	2.35e+03		Std	1.16e+07	2.01e+07	1.80e+07
f_6	Median	**1.63e+01**	**1.61e+01**	1.75e+01	f_6	Median	**1.65e+01**	**1.65e+01**	1.79e+01
	Mean	1.63e+01	1.62e+01	1.74e+01		Mean	1.64e+01	1.65e+01	1.78e+01
	Std	3.10e-01	3.38e-01	3.52e-01		Std	3.56e-01	2.52e-01	3.57e-01
f_7	Median	**3.27e+02**	**6.08e+02**	7.22e+04	f_7	Median	**3.33e+02**	**7.61e+02**	7.87e+05
	Mean	9.52e+02	9.80e+02	7.20e+04		Mean	1.20e+03	1.35e+03	7.85e+05
	Std	1.23e+03	1.06e+03	1.07e+04		Std	1.59e+03	1.80e+03	4.70e+04
f_8	Median	**3.33e+02**	**9.41e+02**	7.42e+05	f_8	Median	**4.04e+02**	9.27e+05	**4.21e+02**
	Mean	1.12e+03	1.95e+03	7.41e+05		Mean	3.20e+05	1.36e+06	1.03e+03
	Std	1.68e+03	2.33e+03	4.02e+04		Std	1.11e+06	1.22e+06	1.23e+03
f_9	Median	3.75e+10	3.65e+10	**3.71e+09**	f_9	Median	1.90e+11	1.63e+11	**1.44e+10**
	Mean	3.92e+10	3.96e+10	3.91e+09		Mean	1.88e+11	1.75e+11	1.65e+10
	Std	1.27e+10	1.24e+10	1.18e+09		Std	4.35e+10	3.14e+10	5.79e+09
f_{10}	Median	2.81e+05	2.84e+05	**1.15e+05**	f_{10}	Median	2.26e+07	2.35e+07	**8.33e+06**
	Mean	2.83e+05	2.83e+05	1.11e+05		Mean	2.31e+07	2.39e+07	8.61e+06
	Std	1.79e+04	1.55e+04	1.95e+04		Std	2.51e+06	2.11e+06	1.44e+06
f_{11}	Median	**1.01e+01**	**1.09e+01**	**1.04e+01**	f_{11}	Median	**1.02e+01**	**1.04e+01**	8.97e+02
	Mean	1.03e+01	1.06e+01	1.07e+01		Mean	1.04e+01	1.03e+01	1.95e+04
	Std	1.03e+00	1.08e+00	9.71e-01		Std	9.68e-01	8.10e-01	4.01e+04
f_{12}	Median	3.55e+05	3.69e+05	**5.21e+04**	f_{12}	Median	2.83e+06	2.96e+06	**4.77e+04**
	Mean	3.81e+05	3.84e+05	5.18e+04		Mean	2.95e+06	3.76e+06	4.80e+04
	Std	9.10e+04	9.03e+04	1.48e+04		Std	9.48e+05	3.43e+06	1.33e+04
f_{13}	Median	2.53e+06	3.77e+05	**1.64e+05**	f_{13}	Median	1.36e+06	1.06e+06	**7.17e+05**
	Mean	2.14e+06	5.62e+05	9.85e+05		Mean	1.44e+06	1.09e+06	8.33e+05
	Std	1.08e+06	4.85e+05	3.03e+06		Std	5.42e+05	3.37e+05	2.00e+05
f_{14}	Median	2.66e+13	2.17e+13	**2.21e+12**	f_{14}	Median	5.61e+12	5.35e+12	**3.12e+11**
	Mean	2.62e+13	2.18e+13	2.31e+12		Mean	6.96e+12	5.99e+12	1.40e+12
	Std	9.01e+12	4.93e+12	8.31e+11		Std	2.92e+12	2.74e+12	5.44e+12
f_{15}	Median	5.58e+07	5.48e+07	**1.99e+07**	f_{15}	Median	1.74e+08	1.73e+08	**6.84e+07**
	Mean	5.61e+07	5.55e+07	2.02e+07		Mean	1.72e+08	1.71e+08	6.56e+07
	Std	3.73e+06	4.50e+06	4.95e+06		Std	1.36e+07	1.34e+07	1.82e+07
f_{16}	Median	**1.13e-08**	**1.21e-08**	9.32e+01	f_{16}	Median	**7.84e-09**	9.69e-09	5.89e+05
	Mean	1.12e-08	1.29e-08	1.40e+04		Mean	7.75e-09	1.06e-08	4.66e+05
	Std	1.07e-09	5.26e-09	3.90e+04		Std	1.56e-09	3.35e-09	3.75e+05
f_{17}	Median	6.38e+09	3.28e+09	**7.05e+01**	f_{17}	Median	5.39e+08	4.47e+08	**1.18e+02**
	Mean	6.50e+09	3.97e+09	1.53e+03		Mean	7.73e+08	5.92e+08	2.45e+02
	Std	1.94e+09	1.96e+09	6.39e+03		Std	3.76e+08	3.26e+08	6.45e+02
f_{18}	Median	7.84e+09	6.39e+09	**2.52e+09**	f_{18}	Median	2.65e+07	1.93e+07	**1.08e+07**
	Mean	9.87e+09	9.23e+09	3.64e+09		Mean	3.28e+07	2.24e+07	1.42e+07
	Std	4.17e+09	4.88e+09	4.06e+09		Std	1.52e+07	1.00e+07	1.21e+07

The highlighted entries are significantly better

decomposition method as mentioned above. The performance of SACC3 deteriorates on the most of functions with one non-separable subcomponent, as mentioned in Section 4.2, but when there are the several non-separable subcomponents with the different imbalances, its results are better than CBCC1 and CBCC2 due to not only considering subcomponent with maximum effect, but also considering all subcomponents' effects.

4.3.2 Experiment series 2: comparison of methods along with the ideal decomposition method

Table 8 shows the obtained results of SACC3, CBCC1, and CBCC2 which use the ideal decomposition method. On the modified CEC-2010 test functions with normal weights, Table 8a indicates SACC3 is the best performing algorithm and outperforms other algorithms on 11 out of 15 functions while CCBC1 and CCBC2 obtain best results for 7 out of 15 functions. It can be seen from Table 8b on the modified CEC-2010 test functions that SACC3 has the best performance among other algorithms on 11 out of 15 functions while CCBC1 and CCBC2 have the best performance on 6 and 4 out of 15 functions, respectively. Table 9 shows the results of CC algorithm with SACC3, CBCC1, and CBCC2 by using the ideal decomposition method on the CEC-2013 LSGO benchmark functions. From Table 9, it is seen SACC3 is the best performing algorithm and outperforms other algorithms on 8 out of 10 functions while CCBC1 and CCBC2 obtain best results for 2 out of 10 functions. From the study of Tables 8 and 9, we can conclude that the performance of SACC3 is either significantly better than or comparable to CBCC1 and CBCC2. An important observation from the results of all tables is that as the number of imbalanced subcomponent increases (f_9–f_{13} and f_{14}–f_{18}) in the modified CEC-2010 test functions and the modified CEC-2010 test functions with normal weights, and imbalance functions f_4–f_{11} and f_{13}–f_{14} in the CEC-2013 LSGO benchmark functions, the performance of SACC3 also improves and outperforms CBCC1 and CBCC2. It is noteworthy that the performance of SACC3 increases significantly according to Tables 8 and 9 when decomposition method can obtain a near optimal grouping of the decision variables. An important reason of the ability SACC3 to find better solutions is that it can allocate computational resources to all subcomponents well according their main effect of the global fitness value. To gain a better understanding of the behavior of algorithms, we plot the convergence graph on six selected problems (two functions from three classes of functions) on the modified CEC-2010 benchmark functions and the modified CEC-2010 test functions with normal weights in Figs. 9 and 10, respectively. Also, the convergence plots for four selected functions on the CEC-2013 LSGO benchmark functions is shown in Fig. 11.

Table 9 Results of SACC3 and contribution based CC algorithm with the ideal decomposition method for the CEC-2013 LSGO benchmark functions

Function		CBCC1	CBCC2	SACC3
f_4	Median	2.60e+10	2.37e+10	**1.60e+10**
	Mean	2.60e+10	2.76e+10	1.81e+10
	Std	1.31e+10	1.43e+10	8.53e+09
f_5	Median	4.40e+06	4.39e+06	**2.35e+06**
	Mean	4.42e+06	4.40e+06	2.33e+06
	Std	4.56e+05	4.24e+05	5.02e+05
f_6	Median	**1.43e+01**	**1.43e+01**	8.15e+04
	Mean	4.07e+03	1.44e+04	7.92e+04
	Std	1.41e+04	3.13e+04	3.62e+04
f_7	Median	4.30e+07	5.22e+07	**1.51e+06**
	Mean	4.58e+07	5.46e+07	1.76e+07
	Std	2.37e+07	2.49e+07	6.23e+07
f_8	Median	2.65e+15	2.29e+15	**1.12e+13**
	Mean	2.87e+15	2.48e+15	1.15e+13
	Std	1.15e+15	9.83e+14	5.78e+12
f_9	Median	4.30e+08	4.23e+08	**1.43e+08**
	Mean	4.30e+08	4.28e+08	1.67e+08
	Std	2.28e+07	4.06e+07	5.93e+07
f_{10}	Median	**4.11e+00**	**3.65e+00**	1.11e+02
	Mean	1.44e+01	8.88e+00	1.06e+02
	Std	1.84e+01	1.05e+01	1.68e+01
f_{11}	Median	1.73e+09	1.76e+09	**1.11e+08**
	Mean	2.76e+09	2.25e+09	1.70e+10
	Std	2.56e+09	1.97e+09	3.86e+10
f_{13}	Median	9.57e+09	7.76e+09	**1.17e+09**
	Mean	8.85e+09	8.46e+09	3.31e+09
	Std	3.06e+09	2.99e+09	3.71e+09
f_{14}	Median	6.87e+10	6.80e+10	**6.08e+08**
	Mean	7.25e+10	7.35e+10	2.81e+10
	Std	2.78e+10	3.16e+10	8.32e+10

The highlighted entries are significantly better

4.4 Comparison among all SACC methods and other three algorithms with multiple tests

4.4.1 Wilcoxon's test

Besides the all above experiments, we also conduct Wilcoxon's test to detect significant differences between the behaviors of algorithms. Each version of the SACC algorithms is compared with the standard round-robin CC,

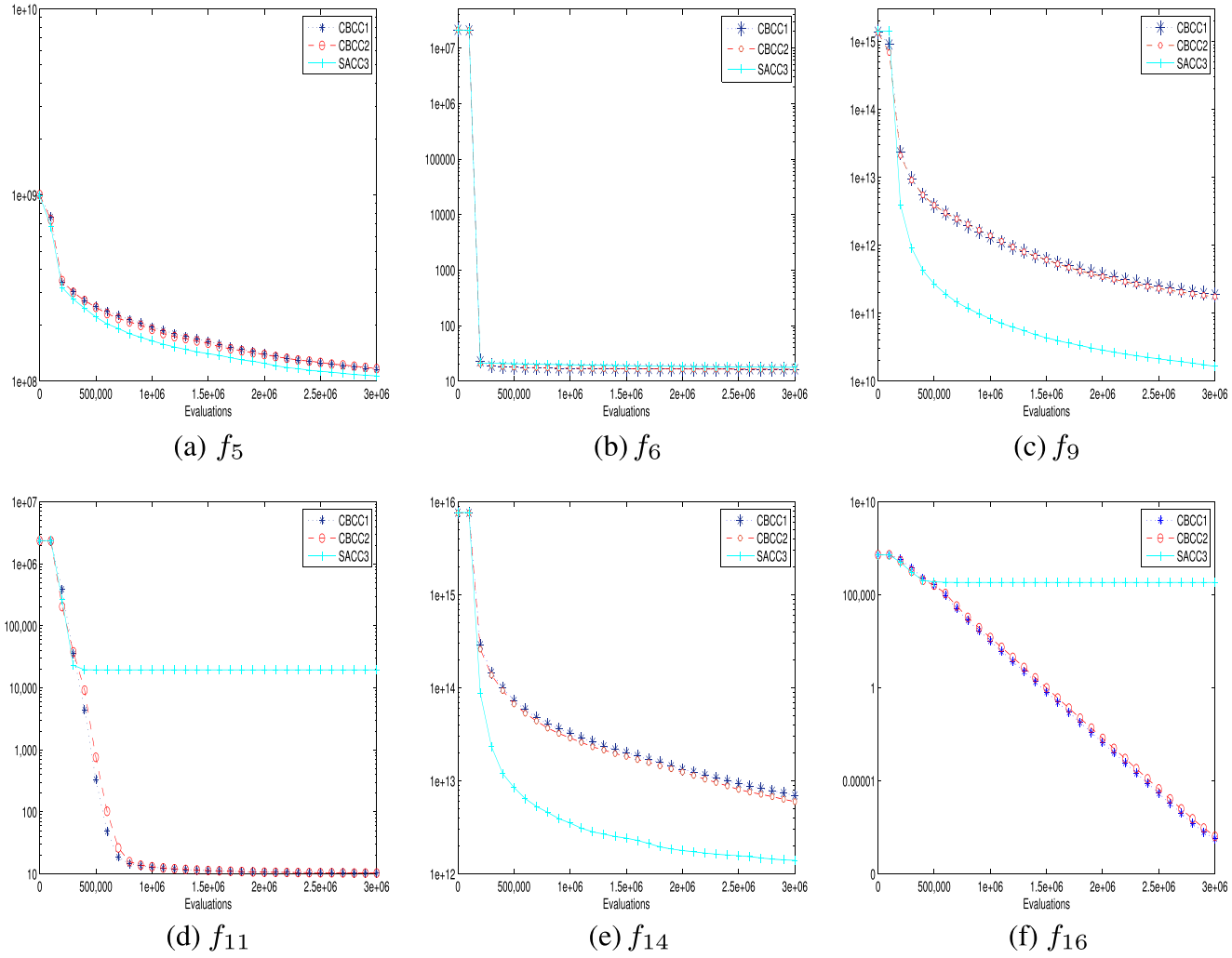


Fig. 9 Convergence plots of f_5, f_6, f_9, f_{11}, f_{14}, and f_{16} of the modified CEC-2010 test functions. The results were averaged over 25 runs. The vertical axis is the function value and the horizontal axis is the number of function evaluations

CBCC1, and CBCC2. Tables 10, 11, 12, 13, and 14 show the ranks and p-values of Wilcoxon test between each version of the SACC algorithms and other three algorithms with the various decomposition methods. A plus sign '+' indicates that SACC algorithms statistically perform better than other algorithms at significance level 5%, whereas a similarity sign '$\sim$' means that no significant differences were detected. From the Tables 10 and 11a, it can be seen that SACC3 is significantly better than other algorithms with the ideal decomposition methods. We observe that the rank values corresponding to SACC3 are always greater than other algorithms in the various decomposition methods. Also, from the Tables 11b and 12 it can be seen that SACC2 is significantly better than the standard round-robin CC algorithm with the ideal decomposition method while both algorithms have no significant differences with the DG decomposition method. The Wilcoxon Signed-Rank test indicates that SACC2 and two versions of CCBC have no significant differences except that SACC2 is significantly better than CBCC1 along with the DG decomposition method. From the Tables 13 and 14, we can observe that SACC1 is significantly better than other algorithms on most cases and the rank values corresponding to SACC1 are always greater than other algorithms with every decomposition method. In addition, the Wilcoxon Signed-Rank test is conducted to detect significant differences between the different versions of SACC. Tables 15 and 16 show the ranks and p-values of Wilcoxon test between versions of SACC algorithm. From the Table 15, we can observe that SACC3 is significantly better than both SACC1 and SACC2 algorithms with the ideal decomposition method and the rank values corresponding to SACC3 are always greater than SACC1 and SACC2 algorithms with every decomposition method. From the Table 16, we can observe that SACC1 is significantly better than SACC2 with the DG decomposition method.

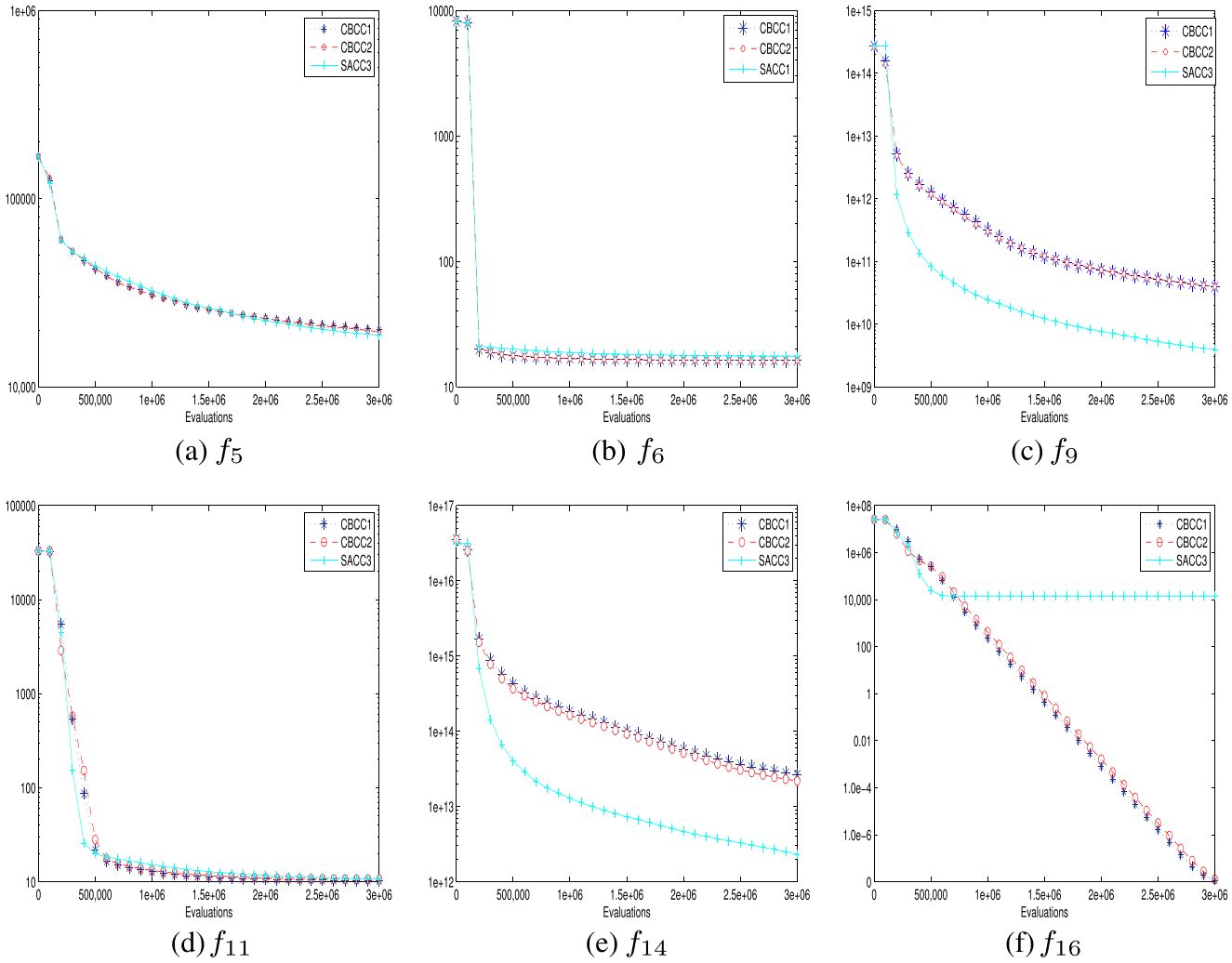


Fig. 10 Convergence plots of f_5, f_6, f_9, f_{11}, f_{14}, and f_{16} of the modified CEC-2010 test functions with normal weights. The results were averaged over 25 runs. The vertical axis is the function value and the horizontal axis is the number of function evaluations

4.4.2 Multiple friedman aligned comparisons with a control method

In addition, Friedman Aligned test and post-hoc procedure are conducted to compare algorithms [6]. The CONTROLTEST software package [5, 8, 9] is used to compute p-values, ranks, and adjusted p-values of these test. Table 17 shows the ranks and related p-values of the Friedman Aligned tests. As we can see from Table 17, the best performing algorithm in the comparison of all algorithms with the DG decomposition method for the Friedman Aligned test is SACC3 (63.233). The p-value computed through the statistics of the Friedman Aligned test is 2.530e-4 which indicates the existence of significant differences among the compared algorithms. It is obvious from Table 17 that in the comparison of all algorithms with the ideal decomposition method, the best performing algorithm by Friedman Aligned test is SACC3 (72.988). The p-value computed through the statistics of the Friedman Aligned test is 7.351e-6. The statistic test confirms significant differences among compared algorithms and indicates that ranks assigned to SACC3 are the best rank. Tables 18 and 19 indicate the eight posthoc procedures (i.e., $Bonf$, $Holm$, $Hoch$, $Homm$, $Holl$, Rom, $Finn$, and Li tests) using the ranks of Friedman Aligned test. In posthoc procedures, the best performing algorithm is highlighted as control algorithm (SACC3) and then all hypotheses of equality among control algorithm and all compared algorithms by computing p-values. For the DG decomposition method, as we can see in Table 18, the Friedman Aligned test confirms the improvement of SACC3 over DECC, CBCC1, and CBCC2 for every post-hoc procedure while it indicates that it behaves similarly with SACC1 and SACC2. Table 19 indicates that for the ideal decomposition method, a significant improvement of SACC3 over all algorithms.

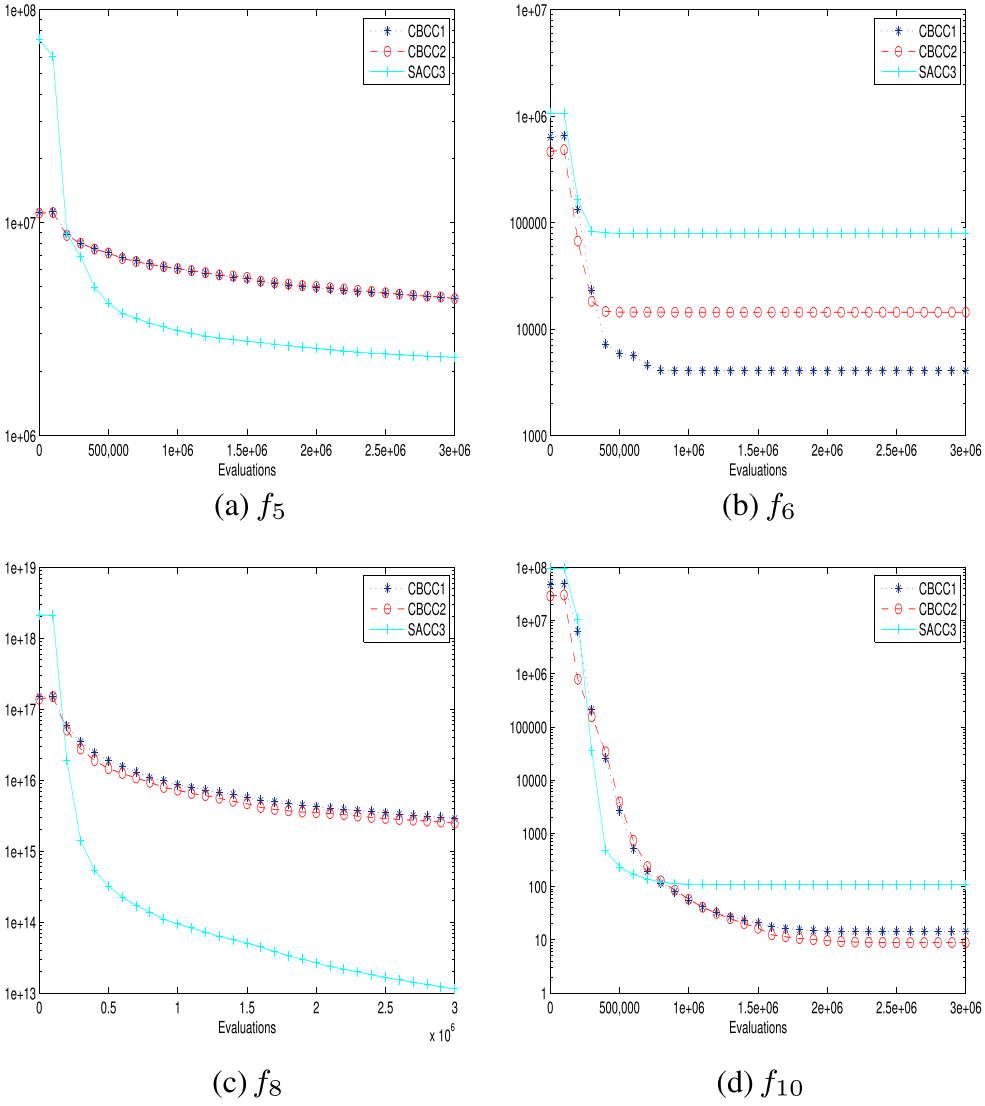


Fig. 11 Convergence plots of f_5, f_6, f_8, and f_{10} of the CEC-2013 LSGO benchmark functions. The results were averaged over 25 runs. The vertical axis is the function value and the horizontal axis is the number of function evaluations

Table 10 Wilcoxon test

Decomposition method	DG	ideal
(a) Wilcoxon test between SACC3 and DECC		
SACC3.Ranks	322	699
DECC.Ranks	143	121
p-value	0.06576	0.0001
Results at the 5% significance level	$\sim$	$+$
(b) Wilcoxon test between SACC3 and CBCC1		
SACC3.Ranks	335	690
CBCC1.Ranks	130	130
p-value	0.03486	0.00016
Results at the 5 % significance level	$+$	$+$

Table 11 Wilcoxon test

Decomposition method	DG	ideal
(a) Wilcoxon test between SACC3 and CBCC2		
SACC3.Ranks	323	672
CBCC2.Ranks	142	148
p-value	0.06288	0.00044
Results at the 5 % significance level	$\sim$	$+$
(b) Wilcoxon test between SACC2 and DECC		
SACC2.Ranks	238	689
DECC.Ranks	113	131
p-value	0.11184	0.00018
Results at the 5 % significance level	$\sim$	$+$

Table 12 Wilcoxon test

Decomposition method	DG	ideal
(a) Wilcoxon test between SACC2 and CBCC1		
SACC2.Ranks	367.5	330.5
CBCC1.Ranks	97.5	372.5
p-value	0.00544	0.74896
Results at the 5 % significance level	+	∼
(b) Wilcoxon test between SACC2 and CBCC2		
SACC2.Ranks	181.5	298
CBCC2.Ranks	224.5	443
p-value	0.62414	0.29372
Results at the 5 % significance level	∼	∼

Table 13 Wilcoxon test between SACC1 and DECC

Decomposition method	DG	ideal
(a) Wilcoxon test between SACC1 and DECC		
SACC1.Ranks	401	618
DECC.Ranks	5	123
p-value	0	0.00034
Results at the 5 % significance level	+	+
(b) Wilcoxon test between SACC1 and CBCC1		
SACC1.Ranks	377	557.5
CBCC1.Ranks	29	222.5
p-value	8e-05	0.01928
Results at the 5 % significance level	+	+

Table 14 Wilcoxon test between SACC1 and CBCC2

Decomposition method	DG	ideal
SACC1.Ranks	282.5	403.5
CBCC2.Ranks	182.5	376.5
p-value	0.30302	0.8493
Results at the 5 % significance level	∼	∼

Table 15 Wilcoxon test

Decomposition method	DG	ideal
(a) Wilcoxon test between SACC3 and SACC1		
SACC3.Ranks	321	672
SACC1.Ranks	144	148
p-value	0.06876	0.00044
Results at the 5 % significance level	∼	+
(b) Wilcoxon test between SACC3 and SACC2		
SACC3.Ranks	321	650
SACC2.Ranks	144	170
p-value	0.06876	0.00017
Results at the 5 % significance level	∼	+

Table 16 Wilcoxon test between SACC1 and SACC2

Decomposition method	DG	ideal
SACC1.Ranks	294	529.5
SACC2.Ranks	112	250.5
p-value	0.03846	0.05118
Results at the 5 % significance level	+	∼

Table 17 Ranks and p-values achieved by the friedman aligned test

Algorithm	Friedman aligned	
Decomposition method	DG	ideal
DECC	104.850	154.363
SACC1	79.233	119.562
SACC2	90.217	125.75
SACC3	63.233	72.988
CBCC1	105.783	129.588
CBCC2	99.683	120.750
p-value	2.530e-4	7.351e-6

Table 18 Adjusted p-values for the Friedman Aligned test (SACC3 is the control algorithm) for the DG decomposition method

i	algorithm	unadjusted p	p_{Bonf}	p_{Holm}	p_{Hoch}	p_{Homm}	p_{Holl}	p_{Rom}	p_{Finn}	p_{Li}
1	CBCC1	0.002	0.008	0.008	0.008	0.006	0.008	0.007	0.008	0.002
2	DECC	0.002	0.010	0.008	0.008	0.008	0.008	0.008	0.008	0.003
3	CBCC2	0.007	0.034	0.0208	0.020	0.020	0.020	0.020	0.011	0.009
4	SACC2	0.045	0.224	0.090	0.090	0.090	0.088	0.090	0.056	0.055
5	SACC1	0.234	1.172	0.234	0.234	0.234	0.234	0.234	0.234	0.234

5 Conclusions and future directions

In this paper, we proposed a CC algorithm with a sensitivity analysis-based budget assignment method to tackle the imbalanced LSGO problems. The Morris screening method has been incorporated into the CC algorithm to enhance its ability for handling the imbalanced large-scale problems. In SACC, first the main effect of each variable is computed by using Morris screening method and then the main effect of each subcomponent is calculated based on the main effect of its variables. SACC1 and SACC2 consider only the subcomponent with the maximum main effect after each cycle of the CC algorithm. This subcomponent is optimized at one iteration in SACC1 and SACC2 optimizes it until the best obtained solution can be improved. In SACC3, a special number of the computational budgets is assigned to all subcomponents according to their main effect. In SACC, the associated optimization iterations of all subcomponents are based on their main effect of the global fitness value at each cycle of the CC algorithm. Experimental results confirmed that SACC is more efficient to assign budget among all subcomponents if the accuracy of decomposition method will be high or near optimal decomposition method. The performance of SACC was evaluated on two different modified CEC-2010 and CEC-2013 LSGO benchmark functions. The experimental results showed that SACC is very effective and efficient in tackling high-dimensional problems including more imbalanced subcomponents. Also, we demonstrated that a proper budget assignment method can greatly enhance the performance of CC algorithms on the imbalanced LSGO problems. Furthermore, SACC was compared with the standard CC algorithms and contribution-based CC algorithms. The performance of SACC is superior to or at least competitive with the compared methods. In the future, we are planning to examine other sensitivity analysis methods to better capture the effect of various subcomponents on the global fitness, especially on the CEC-2013 LSGO benchmark functions which sensitivity analysis methods can hardly approximate effect of variables on these functions. In addition, we are interested in considering the effect of the dimension size of the subcomponents to have a better budget assignment.

Compliance with Ethical Standards

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

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

Appendix

Tables 20, 21, and 22 present the normal coefficient corresponding to non-separable subcomponents in the modified CEC-2010 test functions with normal weights.

Table 20 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 19 Adjusted p-values for the Friedman Aligned test (SACC3 is the control algorithm)for the ideal decomposition method

i	algorithm	unadjusted p	p_{Bonf}	p_{Holm}	p_{Hoch}	p_{Homm}	p_{Holl}	p_{Rom}	p_{Finn}	p_{Li}
1	DECC-I	1.590e-7	7.950e-7	7.950e-7	7.950e-7	7.950e-7	7.950e-7	7.560e-7	7.950e-7	1.594e-7
2	CBCC1	2.664e-4	0.001	0.001	0.001	0.001	0.001	0.001	6.660e-4	2.671e-4
3	SACC2	6.770e-4	0.003	0.002	0.002	0.002	0.002	0.002	0.001	6.784e-4
4	CBCC2	0.002	0.010	0.004	0.003	0.003	0.004	0.003	0.003	0.002
5	SACC1	0.003	0.013	0.004	0.003	0.003	0.004	0.003	0.003	0.003

Table 21 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 22 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
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

References

1. Campolongo F, Cariboni J, Saltelli A (2007) An effective screening design for sensitivity analysis of large models. Environ Model Softw 22(10):1509–1518
2. Chen S, Montgomery J, Bolufé-Röhler A (2015) Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution. Appl Intell 42(3):514–526
3. Chen W, Tang K (2013) Impact of problem decomposition on cooperative coevolution. In: IEEE congress on evolutionary computation (CEC), 2013. IEEE, pp 733–740
4. Chen W, Weise T, Yang Z, Tang K (2010) Large-scale global optimization using cooperative coevolution with variable interaction learning. In: Parallel problem solving from nature, PPSN XI. Springer, pp 300–309
5. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7(Jan):1–30
6. Derrac J, García S, Molina D, Herrera F (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
7. Ekström PA (2005) Eikos: a simulation toolbox for sensitivity analysis in matlab. Uppsala University, Uppsala
8. García S, Fernández A, Luengo J, Herrera F The software for computing the advanced multiple comparison, http://sci2s.ugr.es/sicidm
9. García S, Fernández A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. Inf Sci 180(10):2044–2064
10. Hasanzadeh M, Meybodi MR, Ebadzadeh MM (2013) Adaptive cooperative particle swarm optimizer. Appl Intell 39(2):397–420
11. Li X, Tang K, Omidvar MN, Yang Z, Qin K (2013) Benchmark functions for the cec'2013 special session and competition on large-scale global optimization. Gene 7:33
12. Liu J, Tang K (2013) Scaling up covariance matrix adaptation evolution strategy using cooperative coevolution. In: Intelligent data engineering and automated learning–IDEAL 2013. Springer, pp 350–357
13. Liu Y, Yao X, Zhao Q, Higuchi T (2001) Scaling up fast evolutionary programming with cooperative coevolution. In: Proceedings of the 2001 congress on evolutionary computation, 2001. IEEE, vol 2, pp 1101–1108
14. Mahdavi S, Shiri ME, Rahnamayan S (2014) Cooperative co-evolution with a new decomposition method for large-scale optimization. In: 2014 IEEE congress on evolutionary computation (CEC). IEEE, pp 1285–1292
15. Mahdavi S, Shiri ME, Rahnamayan S (2015) Metaheuristics in large-scale global continues optimization: a survey. Inf Sci 295:407–428
16. Morris MD (1991) Factorial sampling plans for preliminary computational experiments. Technometrics 33(2):161–174
17. Omidvar MN, Li X (2011) A comparative study of CMA-ES on large scale global optimisation. In: AI 2010: advances in artificial intelligence. Springer, pp 303–312
18. Omidvar MN, Li X, Mei Y, Yao X (2014) Cooperative co-evolution with differential grouping for large scale optimization. IEEE Trans Evol Comput 18(3):378–393
19. Omidvar MN, Li X, Tang K (2015) Designing benchmark problems for large-scale continuous optimization. Inf Sci 316:419–436
20. Omidvar MN, Li X, Yang Z, Yao X (2010) Cooperative co-evolution for large scale optimization through more frequent random grouping. In: IEEE congress on evolutionary computation (CEC), 2010. IEEE, pp 1–8
21. Omidvar MN, Li X, Yao X (2010) Cooperative co-evolution with delta grouping for large scale non-separable function optimization. In: IEEE congress on evolutionary computation (CEC), 2010. IEEE, pp 1–8
22. Omidvar MN, Li X, Yao X (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
23. Potter MA (1997) The design and analysis of a computational model of cooperative coevolution. PhD thesis, Citeseer
24. Potter MA, De Jong KA (1994) A cooperative coevolutionary approach to function optimization. In: Parallel problem solving from nature-PPSN III. Springer, pp 249–257

25. Rabitz H, Aliş ÖF (1999) General foundations of high-dimensional model representations. J Math Chem 25(2–3): 197–233
26. Ray T, Yao X (2009) A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning. In: IEEE Congress on evolutionary computation, 2009. CEC'09. IEEE, pp 983–989
27. Saltelli A, Chan K, Scott EM et al. (2000) Sensitivity analysis, vol 134. Wiley, New York
28. Saltelli A, Ratto M, Andres T, Campolongo F, Cariboni J, Gatelli D, Saisana M, Tarantola S (2008) Global sensitivity analysis: the primer. Wiley
29. Sayed E, Essam D, Sarker R (2012) Dependency identification technique for large scale optimization problems. In: IEEE Congress on evolutionary computation (CEC), 2012. IEEE, pp 1–8
30. Sayed E, Essam D, Sarker R (2012) Using hybrid dependency identification with a memetic algorithm for large scale optimization problems. In: Simulated evolution and learning. Springer, pp 168–177
31. Shan S, Wang GG (2010) Metamodeling for high dimensional simulation-based design problems. J Mech Des 132(5):051009
32. Sheskin DJ (2003) Handbook of parametric and nonparametric statistical procedures. CRC Press
33. Shi Y-j, Teng H-f, Li Z-q (2005) Cooperative co-evolutionary differential evolution for function optimization. In: Advances in natural computation. Springer, pp 1080–1088
34. Singh HK, Ray T (2010) Divide and conquer in coevolution: a difficult balancing act. In: Agent-based evolutionary search. Springer, pp 117–138
35. Tang K, Li X, Suganthan PN, Yang Z, Weise T (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
36. Tenne Y, Goh C-K (2010) Computational intelligence in expensive optimization problems, vol 2. Springer
37. Van den Bergh F, Engelbrecht AP (2004) A cooperative approach to particle swarm optimization. IEEE Trans Evol Comput 8(3):225–239
38. Weise T, Chiong R, Tang K (2012) Evolutionary optimization: pitfalls and booby traps. J Comput Sci Technol 27(5):907–936
39. Yang Z, Tang K, Yao X (2008) Large scale evolutionary optimization using cooperative coevolution. Inf Sci 178(15):2985–2999
40. Yang Z, Tang K, Yao X (2008) Multilevel cooperative coevolution for large scale optimization. In: IEEE Congress on evolutionary computation, 2008. CEC 2008. (IEEE world congress on computational intelligence). IEEE, pp 1663–1670
41. Yang Z, Tang K, Yao X (2008) Self-adaptive differential evolution with neighborhood search. In: IEEE Congress on evolutionary computation, 2008. CEC 2008. (IEEE world congress on computational intelligence). IEEE, pp 1110–1116