Applied Soft Computing Journal 92 (2020) 106295

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

A topology-based single-pool decomposition framework for large-scale global optimization

Xiaoming Xue [a], Kai Zhang [a,*], Rupeng Li [b], Liming Zhang [a], Chuanjin Yao [a], Jian Wang [c], Jun Yao [a]

[a] School of Petroleum Engineering, China University of Petroleum (East China), Qingdao 266580, PR China
[b] School of Minerals and Energy Resources Engineering, University of New South Wales, Sydney 2052, Australia
[c] College of Science, China University of Petroleum (East China), Qingdao 266580, PR China

ARTICLE INFO

Article history:
Received 11 August 2019
Received in revised form 13 January 2020
Accepted 5 April 2020
Available online 16 April 2020

Keywords:
Large-scale global optimization
Problem decomposition
Cooperative coevolution
Topology information

ABSTRACT

Identification of variable interaction plays a crucial role in applying a divide-and-conquer algorithm for large-scale black-box optimization. However, most of the existing decomposition methods are less efficient in decomposing the overlapping problems. This drawback diminishes the practicality of the existing methods. In this paper, we propose an efficient single-pool decomposition framework (SPDF). The interactions of decision variables are identified in an ordinal fashion. The unbalanced grouping efficiency of the existing decomposition methods can be significantly alleviated. Furthermore, we find that the grouping efficiency can be further improved by integrating the topological information into the decomposition process. In many real-world problems, this information can be 1-, 2- or 3-dimensional coordinates, which represent the geometric structure of the large-scale systems. Based on this, we propose a topology-based decomposition method, which we call Topology-based Single-Pool Differential Grouping (TSPDG). The efficacy of our proposed methods is demonstrated on the CEC'2010 and the CEC'2013 large-scale benchmark suites, as well as a practical case study in production optimization.

1. Introduction

In recent years, there are increasing number of large-scale global optimization (LSGO) problems in the real world, arising in domains spanning science and engineering fields [1–4]. At the same time, the number of decision variables has shown an exponential increase for most engineering problems [5]. Aerodynamic shape optimization [6], seismic waveform inversion [7], and waterflooding optimization in reservoir management [8,9] are just a few examples among various large-scale optimization problems.

The main contributing factors that make LSGO problems extremely complex are that the search space of an optimization problem grows exponentially and the landscape of fitness function may change a lot with the dimensionality increasing. For example, the Rosenbrock function is unimodal in two dimensions, but it becomes multimodal when the dimension is larger than four [10,11]. How to address this challenge effectively has become an urgent problem. A large number of scalable algorithms have been proposed and studied in the evolutionary computation [12, 13] and classic mathematical programming domain [14,15]. However, the evaluation of large-scale problems is usually time consuming especially on engineering optimization problems, such as multidisciplinary design optimization [16], waterflooding optimization [8], and target shape design optimization [6].

There are three popular methods can be used to solve LSGO problems, i.e., exploration enhancement [17], dimensionality reduction [18] and divide-and conquer (a.k.a decomposition) methods [19], among which decomposition methods have gained increasingly attention in recent years. The cooperative co-evolution (CC) method, proposed by Potter and De Jong in [20], decomposes the original large-scale problem into a certain number of one-dimensional sub-problems before the evolution. Each subproblem maintains a subpopulation which can be evolved by a subcomponent optimizer in a round-robin fashion. In conjunction with different evolutionary strategies, a series of variants have been developed under the traditional CC [13]. Such as the cooperative particle swarm optimizer (CPSO) [21], covariance matrix adaptation evolution strategy with CC (CC-CMA-ES) [22], and fast evolutionary programming with CC (FEPCC) [23], etc. However, when dealing with LSGO problems, the strong interdependency among the subcomponents [23–27] deteriorate the performance

* Corresponding author.
E-mail address: zhangkai@upc.edu.cn (K. Zhang).

https://doi.org/10.1016/j.asoc.2020.106295
1568-4946/

of CC dramatically. Therefore, the identification of variable interaction (i.e., problem decomposition) plays a crucial role in applying CC methods.

Most decomposition strategies can be classified into two categories: static decomposition and dynamic decomposition. In the static decomposition strategies (e.g. Differential Grouping [28]), the decomposition procedure is performed before the optimization with CC framework. Recently, a series of variants of differential grouping have been proposed to decompose the LSGO problems. These sophisticated decomposition methods have shown superior performance in decomposing most of the functions on the latest large-scale continuous optimization benchmark suites [29, 30]. In the second method, *Dynamic Decomposition*, the interaction structure is identified during the optimization process (e.g. Delta Grouping [31]). However, most of these types of decomposition methods are less effective in dealing with partially separable functions.

To the best of our knowledge, the highest dimension of computationally expensive problems ever solved by surrogate-assisted heuristic algorithms is 100 [32]. A divide-and-conquer approach for dealing with large-scale optimization problems, cooperative co-evolution assisted by surrogate is promising to solve large-scale time-consuming optimization problems up to a dimension of 1000 [32]. Multiple low-dimensional surrogates can be conducted for the subproblems and be optimized in a cooperative coevolutionary fashion. However, it is well-known that the computational budgets of the time-consuming problems are always limited. An inefficient decomposition method may impede the application of the cooperative coevolution in solving such problems. Therefore, a more efficient decomposition method is needed.

Fast Interdependency Identification (FII) [33], Differential Grouping 2 (DG2) [34], and Recursive Differential Grouping (RDG) [35], published recently, are a few competitive decomposition methods that can identify the nonseparable subcomponents of an LSGO problem and have shown superior performance as compared to other decomposition algorithms such as variable interaction learning [27] on the latest large-scale benchmark suites. However, all above methods have the following major shortcomings:

- High or unbalanced computational complexity on large-scale optimization problems. When decomposing an n-dimensional problem, the computational complexity of DG2 is $\mathcal{O}(n^2)$, the computational costs of FII and RDG are affected by the variable interaction structure of the given problems. With these two decomposition methods, the overlapping functions consume considerably more computational resources than other forms of functions.
- The topological information of many large-scale real-world problems are not taken into account by these methods. However, these information may be useful to significantly improve the grouping efficiency.

In this paper, we firstly revisit the theorem of interaction identification and prove it from a new point of view. This helps us to obtain a deeper understanding of the theorem of interaction identification. Then, a Single-Pool Decomposition Framework (SPDF) is proposed in this paper. The interactions of decision variables can be identified one by one. Based on the SPDF, an efficient and accurate decomposition algorithm for the additively separable problems named Single-Pool Differential Grouping (SPDG) is presented. In order to further improve the decomposition efficiency, a topology-based interaction identification strategy is proposed in this paper. For problems with numerous topological decision variables, the proposed Topology-based Single-Pool Differential Grouping (TSPDG) shows superior decomposition performance with respect to the grouping efficiency. The experimental results on the CEC'2010 and CEC'2013 benchmark suites verify the effectiveness of the proposed methods.

The rest of this paper is organized as follows. Section 2 introduces several basic notions in decomposition-based CC architecture and also briefly reviews the related work. Section 3 contains the detail of proposed single-pool decomposition framework and topological decomposition strategy. The experimental results about the decomposition performance of SPDF-based algorithms are presented in Section 4. At last, Section 5 summarizes and concludes the paper.

2. Related work

2.1. Separable function

Definition 1 (*[36]*)**.** An optimization function $f(\mathrm{x})$ is partially separable with m independent components iff:

$$\arg\min_{\mathrm{x}} f(\mathrm{x}) = \left(\arg\min_{\mathrm{x}_1} f\left(\mathrm{x}_1, \ldots\right), \ldots, \arg\min_{\mathrm{x}_m} f\left(\ldots, \mathrm{x}_m\right) \right)$$

where $\mathrm{x} = (x_1, \ldots, x_n)^\top$ is a decision vector whose decision space $S \in \mathbb{R}^n$, $\mathrm{x}_1, \ldots, \mathrm{x}_m$ are disjoint sub-vectors of x.

In particular, $f(\mathrm{x})$ is fully separable when $m = n$. In this case, there are no interaction between any pair of variables. This kind of problem can be solved by optimizing each of the decision variables independently. $f(\mathrm{x})$ is fully non-separable when $m = 1$. Note that all decision variables of fully nonseparable problem should be optimized together [36].

Generally speaking, it is a challenging task to obtain a LSGO's interaction information due to its black-box characteristic [37]. To the best of our knowledge, there is not yet an effective algorithm that can identify the interaction structure of the generally separable problems (i.e., Definition 1). Additively separable functions are a special type of partially separable functions, which can represent the modular nature of many real-world optimization problems [38]. An additively separable function is easier to solve for CC, because its variable interactions are easier to be identified [28]. Such function can be defined as follows:

Definition 2 (*[36]*)**.** A function is partially additively separable if it has the following general form:

$$f(\mathrm{x}) = \sum_{i=1}^{m} f_i(\mathrm{x}_i),\ m > 1 \tag{1}$$

where $f_i(\cdot)$ is a non-separable subfunction, and m is the number of non-separable components of f. The definition of x and x_i is identical to what was given in Definition 1.

An intuitive illustration of additive separability, general separability and non-separability is shown in Fig. 1. The red and blue lines represent the unperturbed and perturbed fitness landscapes of $f_i(x_1, \cdot)$ respectively. The three types of functions are denoted by f_a, f_g and f_n. We can see that the two landscapes of additively separable function are exactly the same with a suitable translation along the y axis. In this case, Δ_1 is equal to Δ_2. It is noteworthy that most of existing decomposition methods are based on this equation. Unlike the additive separability, general separability only guarantees that the optimal solution remain the same despite the perturbation (as shown in the middle of Fig. 1). The equality between Δ_1 and Δ_2 is likely to disappear. This can be used to explain why the general separability is much more challenging to be identified as compared to the additive separability. As for non-separability, the consistency of optimal solution are no longer satisfied, which is shown in the right part of Fig. 1. In this paper, we only focus on the study of problem decomposition for additive separability.

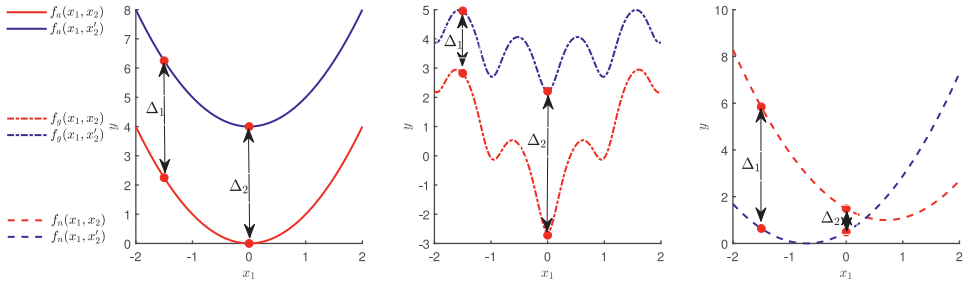


Fig. 1. An intuitive illustration of additive separability (left), general separability (middle) and non-separability (right).

2.2. Problem decomposition

A great number of decomposition algorithms have been proposed to decompose the original LSGO problem into some smaller subproblems. In this paper, we classify a series of decomposition algorithms into two categories, i.e., static decomposition and dynamic decomposition. Several popular algorithms are introduced as follows.

2.2.1. Dynamic decomposition

In dynamic decomposition approaches, variable interactions are identified during the optimization process. The most typical dynamic decomposition algorithm is the random grouping [19] which randomly assigns all variables to a certain number of groups. Though a number of algorithms based on random grouping (e.g., multilevel cooperative co-evolution (MLCC) [39] and more frequent random grouping [40]) have made some improvements, all these stochastic grouping approaches are unable to make near-optimal decomposition for problems with many nonseparable variables [28]. Other dynamic decomposition methods, such as Delta grouping [31] and CCEA-AVP [41] use the historical information to explore the objective function's interaction structure. However, Delta grouping may lose efficacy on those partially separable functions with more than one nonseparable subcomponent [28]. As for CCEA-AVP, it is only effective in detecting linear variable dependencies without nonlinear interdependency despite heavy computational cost [12,42].

2.2.2. Static decomposition

In static decomposition strategies, the decomposition is completed before the execution of CC. The static decomposition strategies are further divided into two categories: detection-based grouping method and fixed-size grouping method. Fixed-size grouping method (e.g., CCGA by Potter et al. [20], divide-in-half method by Shi et al. [43] and the method proposed by van den Bergh and Engelbrecht [21]) divides an n-dimensional optimization problem into m k-dimensional subproblems using presupposed information. However, the performance of these methods may decrease dramatically when meet the problems with strong interdependency among subcomponents. To avoid the issue mentioned above, various detection-based grouping methods (e.g., variable interaction learning (CCVIL) [27], differential grouping (DG) [28], differential grouping 2 (DG2) [34], fast interdependency identification (FII) [33], and recursive differential grouping (RDG) [35]), which do not presuppose the interaction information of variables, were proposed. Among these algorithms, the variants of DG have shown superior performance in terms of decomposition efficiency and accuracy.

The criterion about interdependency identification of DG is as follows:

Theorem 1 ([28]). *Let $f(\mathrm{x})$ be an additively separable function, $\forall a, b_1 \neq b_2, \delta \neq 0$, variables x_p and x_q interact if the following condition holds*

$$\Delta_{\delta,x_p}[f](\mathrm{x})|_{x_p=a,x_q=b_1} \neq \Delta_{\delta,x_p}[f](\mathrm{x})|_{x_p=a,x_q=b_2} \tag{2}$$

where

$$\Delta_{\delta,x_p}[f](\mathrm{x}) = f(\ldots, x_p+\delta, \ldots) - f(\ldots, x_p, \ldots)$$

refers to the forward difference of f with respect to variable x_p with the interval δ.

Theorem 1 states that two variables x_p and x_q interact if equation (2) holds. The proof of Theorem 1 can be found in [28]. For the sake of brevity, the left hand side of Eq. (2) is denoted by $\Delta^{(1)}$ and its right hand side by $\Delta^{(2)}$. It is clear that $\Delta^{(1)} \neq \Delta^{(2)} \iff \lambda = |\Delta^{(1)} - \Delta^{(2)}| > 0$. DG algorithm considers the limited precision of floating-point numbers, by introducing a control parameter ε to convert the above inequality check into a new form $\lambda = |\Delta^{(1)} - \Delta^{(2)}| > \varepsilon$. This control parameter could be used to determine the sensitivity of DG to variable interactions [28].

There are four major drawbacks of DG method. Firstly, it requires a large number of function evaluations especially when the LSGO problem is fully separable [28]. Secondly, it is sensitive to the control parameter ε [34]. Thirdly, DG is unable to detect the indirect nonseparability [44], i.e. overlapping subcomponents. A number of variants of differential grouping were proposed to overcome the shortcomings of DG, such as extended differential grouping (XDG) [44], global differential grouping (GDG) [37], etc. Targeting at the low grouping accuracy issue of the differential grouping, XDG primarily focuses on identifying overlapping functions like Rosenbrock function [10]. GDG addresses the sensitivity issue of DG by introducing an adaptive global control parameter to detect all interactions, and it examines all pairs of variables to detect overlapping subcomponents. However, XDG holds the sensitivity issue of DG and the decomposition performance of GDG deteriorates when dealing with imbalanced functions.

The differential grouping 2 (DG2) estimates a suitable threshold value for each pair of variables by analyzing the magnitude of round off errors and reduces the computational resources by reusing the sample point generated for identifying interactions of variables. The decomposition accuracy of DG2 could reach almost 100% for most of the additively separable functions on the large-scale continuous optimization benchmark suites [29,30]. It has been shown that the minimal number of FEs to identify the complete interaction structure matrix is $(n^2+n+2)/2$ [34]. However, it may not need the complete interaction matrix to identify the connected variables (sub-components) [35]. For example, if decision variable x_1 interacts with x_2 and x_3, the interaction between x_2 and x_3 needs not to be checked, as they belong to the same subcomponent.

Subsequently, FII identifies the interdependency between a variable and a group by introducing a perturbation vector. It

can further improve the decomposition efficiency by avoiding identifying the complete interaction matrix. The criterion of interdependency identification of FII is as follows:

Theorem 2 (*[33]*). *Let $f(\mathrm{x})$ be an additively separable function; $x_p \subset \mathrm{x}$ and $\mathrm{x}_q \subset \mathrm{x}$ be subsets of decision variables: card$(x_p) = 1$ and $x_p \cap \mathrm{x}_q = \varnothing$. $\forall\, a, \boldsymbol{b}, \delta \neq 0$, variable x_p and variables in x_q interact if the following condition holds*

$$\Delta_{\delta,x_p}[f](\mathrm{x})|_{x_p=a,\mathrm{x}_q=\boldsymbol{b}} \neq \Delta_{\delta,x_p}[f](\mathrm{x})|_{x_p=a,\mathrm{x}_q=\boldsymbol{b}+\delta} \tag{3}$$

FII firstly identifies the separable variables by examining the interaction between individual decision variable and the other decision variables. If nonseparable variables are found, a selected decision variable x_i is used to form a subcomponent by examining the interaction between x_i and the remaining nonseparable variables sequentially. This process is carried out recursively until all nonseparable variables have been placed into the subcomponents. The FII method is efficient when used to decompose benchmark problems with a large portion of separable variables or complete direct nonseparability. However on benchmark problems with complete indirect nonseparability (e.g. Rosenbrock function f_{18} on CEC'2010 benchmark suite), the FEs required by FII may still be in the magnitude of $n^2\left(\Theta\left(n^2\right)\right)$ [35]. This drawback significantly diminishes the practicality of the FII method.

To address the issue about computational complexity of the FII, a recursive decomposition method named recursive differential grouping (RDG) was proposed. It examines the interaction between x_i and the remaining nonseparable variables recursively. When decomposing an n-dimensional nonseparable overlapping problem (e.g. Rosenbrock function), the total number of FEs required by RDG is about $6n\log_2(n)$. Moreover, the interdependency identification criterion was generalized to arbitrary sets in RDG. This criterion is as follows:

Theorem 3 (*[35]*). *Let $f(\mathrm{x})$ be an additively separable function; $\mathrm{x}_p \subset \mathrm{x}$ and $\mathrm{x}_q \subset \mathrm{x}$ be subsets of decision variables: $\mathrm{x}_p \cap \mathrm{x}_q = \varnothing$. $\forall\, \boldsymbol{a}, \boldsymbol{b}, \delta \neq 0$, there is some interaction between variables in x_p and x_q if the following condition holds*

$$\Delta_{\delta,\mathrm{x}_p}[f](\mathrm{x})|_{\mathrm{x}_p=\boldsymbol{a},\mathrm{x}_q=\boldsymbol{b}} \neq \Delta_{\delta,\mathrm{x}_p}[f](\mathrm{x})|_{\mathrm{x}_p=\boldsymbol{a},\mathrm{x}_q=\boldsymbol{b}+\delta} \tag{4}$$

where

$$\Delta_{\delta,\mathrm{x}_p}[f](\mathrm{x}) = f\left(\ldots,\mathrm{x}_p+\delta,\ldots\right) - f\left(\ldots,\mathrm{x}_p,\ldots\right)$$

refers to the forward difference of f with respect to variables in x_p with the interval δ.

Recently, two improved versions of RDG were proposed to enhance the grouping performance further. RDG2 [45] adopts the error analysis of DG2 to estimate an adaptive threshold for interaction identification, while RDG3 [46] is adept at dealing with overlapping problems. It can be seen that all above decomposition methods neglect the topology information of decision variables of large-scale problems. However, this information may be useful to significantly speed up the problem decomposition. This issue is discussed further in Section 3.3.

3. A topology-based single-pool decomposition framework

In this section, we firstly prove Theorem 3 from a new point of view. Then, the proposed single-pool decomposition framework is described in details. Lastly, a topology-based decomposition strategy is proposed, which can be easily embedded into the proposed decomposition framework.

3.1. Revisiting the theorem of interaction identification

Theorem 3 can be used to identify the interaction between two subcomponents with arbitrary size. Originally, this Theorem is proved with the help of line integral [35]. Here, we will revisit the Theorem 3 from a new point of view. An equivalent form of Eq. (4) is presented as follows:

$$\Delta_{\delta,\mathrm{x}_p,\mathrm{x}_q}[f](\mathrm{x})|_{\boldsymbol{a},\boldsymbol{b}} \neq \Delta_{\delta,\mathrm{x}_p}[f](\mathrm{x})|_{\boldsymbol{a},\boldsymbol{b}} + \Delta_{\delta,\mathrm{x}_q}[f](\mathrm{x})|_{\boldsymbol{a},\boldsymbol{b}} \tag{5}$$

Three differentials $\Delta_{\delta,\mathrm{x}_p}$, $\Delta_{\delta,\mathrm{x}_q}$, and $\Delta_{\delta,\mathrm{x}_p,\mathrm{x}_q}$ are denoted as:

$$\begin{cases} \Delta_{\delta,\mathrm{x}_p} = f\left(\ldots,\mathrm{x}_p+\delta,\mathrm{x}_q,\ldots\right) - f\left(\ldots,\mathrm{x}_p,\mathrm{x}_q,\ldots\right) \\ \Delta_{\delta,\mathrm{x}_q} = f\left(\ldots,\mathrm{x}_p,\mathrm{x}_q+\delta,\ldots\right) - f\left(\ldots,\mathrm{x}_p,\mathrm{x}_q,\ldots\right) \\ \Delta_{\delta,\mathrm{x}_p,\mathrm{x}_q} = f\left(\ldots,\mathrm{x}_p+\delta,\mathrm{x}_q+\delta,\ldots\right) - f\left(\ldots,\mathrm{x}_p,\mathrm{x}_q,\ldots\right) \end{cases} \tag{6}$$

where $\Delta_{\delta,\mathrm{x}_p}$ and $\Delta_{\delta,\mathrm{x}_q}$ represent the differentials of f with respect to x_p and x_q respectively, and $\Delta_{\delta,\mathrm{x}_p,\mathrm{x}_q}$ is the total differential of f with respect to $\mathrm{x}_p \cup \mathrm{x}_q$.

The Eq. (5) fits the divide-and-conquer strategy of CC intuitively. If subcomponent x_p interacts with x_q, the differentials caused by separated perturbation and union perturbation are different (i.e. $\Delta_{\delta,\mathrm{x}_p,\mathrm{x}_q} \neq \Delta_{\delta,\mathrm{x}_p} + \Delta_{\delta,\mathrm{x}_q}$). Therefore, these two subcomponents should be optimized together. A similar interpretation can be found in [47]. From this point of view, Theorem 3 can be easily derived as follows. A lemma is naturally derived according to additively separable functions' structure.

Lemma 1. *If $f(\mathrm{x})$ is additively separable, then for any $\mathrm{x}_p \subset \mathrm{x}$ we have*

$$\Delta_{\delta,\mathrm{x}_p}[f](\mathrm{x}) = \sum_{i\in P} \Delta_{\delta,\mathrm{x}_{pi}}[f_i](\mathrm{x}_i) \tag{7}$$

where P represents the subset of subfunctions which covers all variables in x_p, and x_{pi} is the subvector of x_p with respect to subfunction i.

Before we move on, Lemma 1 is proved herein. The left term of Eq. (7) is expanded as follows:

$$\Delta_{\delta,\mathrm{x}_p}[f](\mathrm{x}) = f\left(\ldots,\mathrm{x}_p+\delta,\ldots\right) - f\left(\ldots,\mathrm{x}_p,\ldots\right) \tag{8}$$

According to Definition 2, an additively separable function is expanded as follows:

$$f(\mathrm{x}) = \sum_{i=1}^{m} f_i(\mathrm{x}_i) = \sum_{j\in P} f_j\left(\mathrm{x}_j\right) + \sum_{k\in P'} f_k\left(\mathrm{x}_k\right) \tag{9}$$

where P' denotes a complementary set of the subset P in subfunction sets.

Based on Eq. (9), the expanded terms in Eq. (8) are rewritten as follows:

$$f\left(\ldots,\mathrm{x}_p+\delta,\ldots\right) = \sum_{j\in P} f_j\left(\ldots,\mathrm{x}_{pj}+\delta,\ldots\right) + \sum_{k\in P'} f_k\left(\mathrm{x}_k\right)$$

$$f\left(\ldots,\mathrm{x}_p,\ldots\right) = \sum_{j\in P} f_j\left(\mathrm{x}_j\right) + \sum_{k\in P'} f_k\left(\mathrm{x}_k\right)$$

Now, Eq. (7) in Lemma 1 can be easily obtained by substituting the above expansions into Eq. (8), which is shown as follows:

$$\begin{aligned} \Delta_{\delta,\mathrm{x}_p}[f](\mathrm{x}) &= \sum_{i\in P} f_i\left(\ldots,\mathrm{x}_{pi}+\delta,\ldots\right) - \sum_{i\in P} f_i\left(\mathrm{x}_i\right) \\ &= \sum_{i\in P} \Delta_{\delta,\mathrm{x}_{pi}}[f_i](\mathrm{x}_i) \end{aligned} \tag{10}$$

Next, with the help of Lemma 1, Theorem 3 can be easily proved. Assuming that $f(\mathrm{x})$ is an additively separable function, two subcomponents $\mathrm{x}_p \subset \mathrm{x}$ and $\mathrm{x}_q \subset \mathrm{x}$ are separable, then $f(\mathrm{x})$ can be rewritten as follows:

$$\begin{aligned} f(\mathrm{x}) &= \sum_{i=1}^{m} f_i(\mathrm{x}_i) \\ &= \sum_{i\in P} f_i(\mathrm{x}_i) + \sum_{j\in Q} f_j(\mathrm{x}_j) + \sum_{k\in R} f_k(\mathrm{x}_k) \end{aligned} \tag{11}$$

where the first two items represent the subfunctions that cover all variables in x_p and x_q respectively, and the last item represents the rest of subfunctions in $f(\mathrm{x})$.

According to Lemma 1 and Eq. (11), the three differentials of function value can be calculated as follows:

$$\Delta_{\delta,\mathrm{x}_p,\mathrm{x}_q}[f](\mathrm{x}) = \sum_{i\in P} \Delta_{\delta,\mathrm{x}_{pi}}[f_i](\mathrm{x}_i) + \sum_{j\in Q} \Delta_{\delta,\mathrm{x}_{qj}}[f_j](\mathrm{x}_j)$$

$$\Delta_{\delta,\mathrm{x}_p}[f](\mathrm{x}) = \sum_{i\in P} \Delta_{\delta,\mathrm{x}_{pi}}[f_i](\mathrm{x}_i)$$

$$\Delta_{\delta,\mathrm{x}_q}[f](\mathrm{x}) = \sum_{j\in Q} \Delta_{\delta,\mathrm{x}_{qj}}[f_j](\mathrm{x}_j)$$

Then, we have

$$\Delta_{\delta,\mathrm{x}_p,\mathrm{x}_q}[f](\mathrm{x}) = \Delta_{\delta,\mathrm{x}_p}[f](\mathrm{x}) + \Delta_{\delta,\mathrm{x}_q}[f](\mathrm{x})$$

The advantage of revisiting the theorem of interaction identification is that we can comprehend Theorems 1 to 3 from a consistent view. As can be observed from Eq. (6), there are totally four component items used to identify the interaction between two subsets of decision variables. Theoretically, the distinction between Theorems 1 to 3 can be attributed to the difference of subset sizes. The subset patterns of Theorems 1 to 3 are variable–variable, variable–group, and group–group, respectively. A variable can be regarded as a group with only one element. Therefore, Theorem 3 can be seen as a generalized form of Theorems 1 and 2. Herein, for the sake of brevity, the four component items involved in Eq. (6) are denoted as follows:

$$\begin{cases} f_{p,q} \sim f(\ldots, \mathrm{x}_p, \mathrm{x}_q, \ldots) \\ f_{p+\delta,q} \sim f(\ldots, \mathrm{x}_p + \delta, \mathrm{x}_q, \ldots) \\ f_{p,q+\delta} \sim f(\ldots, \mathrm{x}_p, \mathrm{x}_q + \delta, \ldots) \\ f_{p+\delta,q+\delta} \sim f(\ldots, \mathrm{x}_p + \delta, \mathrm{x}_q + \delta, \ldots) \end{cases} \tag{12}$$

Herein, $f_{p,q}$ and $f_{p+\delta,q+\delta}$ are named as reference term with no perturbation and perturbation term with union perturbation, respectively, while $f_{p+\delta,q}$ and $f_{p,q+\delta}$ are named as perturbation terms with individual perturbation. The three types of perturbation terms established here fit the divide-and-conquer strategy intuitively. As mentioned in Section 2.2, DG2 proves that the minimal number of FEs to identify the complete interaction structure matrix is $(n^2+n+2)/2$. It is worth mentioning that the proof is based on the mathematical induction. In this study, the above minimal number of FEs can be obtained easily from the new point of view. In order to identify the entire variable interaction matrix, the reference term $f(\ldots, x_i, x_j, \ldots)$ needs to be evaluated only once, the individual perturbation term $f(\ldots, x_i+\delta, x_j, \ldots)$ needs to be evaluated n times, and the union perturbation term $f(\ldots, x_i+\delta, x_j+\delta, \ldots)$ needs to be evaluated $n(n-1)/2$ times. Therefore, the minimal number of FEs is $1+n+n(n-1)/2 = (n^2+n+2)/2$. Benefiting from the new proof, the computational complexity for identifying the complete interaction relationship can be easily estimated.

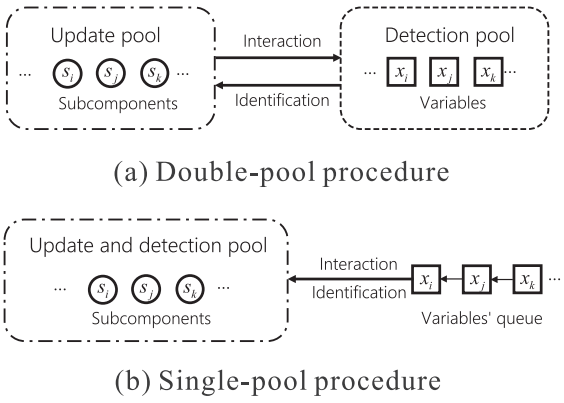


Fig. 2. Double-pool and Single-pool decomposition framework for interaction identification.

3.2. Single-pool decomposition framework

To the best of our knowledge, the existing state-of-the-art static decomposition methods divide all decision variables into two pools: update pool and detection pool. Then, the interactions between subcomponents in update pool and variables in detection pool are dynamically identified. This process is repeatedly executed until there is no variables in the detection pool. The schematic of the above procedure is shown in Fig. 2(a).

As mentioned in Section 2, we can find that FII sequentially identifies the interactions in detection pool and then updates the subcomponents in update pool. In contrast, RDG detects the interactions in a recursive manner. However, this double-pools framework performs poorly in several problems, especially in the overlapping functions (e.g., Rosenbrock function). Thus, a more robust single-pool decomposition framework (SPDF) is proposed in this paper. As Fig. 2(b) shows, the original two pools are integrated into a single one in the new framework. The update and the detection objects are both subcomponents. In this case, the interactions of decision variables can be identified one by one. Algorithm 1 shows the high-level structure of SPDF.

Algorithm 1: *Subcomponents* = SPDF(f, n)

```
The first subcomponent G_1 is set to {1};
for i = 2 → n do
    if There are interactions between variable i and the subcomponents then
        Variable i and the interacted subcomponents are merged together;
    else
        Variable i is set to be a new subcomponent;
```

As mentioned earlier, additively separable functions are a special type of partially separable functions. This type of problem is easier to solve for CCEAs, because its variable interactions are easier to be identified. Among various interaction identification algorithms, perturbation-based methods have shown superior performance with respect to grouping efficiency and accuracy. In this paper, variable-subcomponent interaction identification theorem (i.e., Theorem 2) is selected to be embedded into the proposed SPDF framework. Algorithm 2 shows the pseudo code of Single-Pool Differential Grouping (SPDG). For the sake of brevity the $f(\ldots, x_p, \mathrm{x}_q)$, $f(\ldots, x_p+\delta, \mathrm{x}_q)$, $f(\ldots, x_p, \mathrm{x}_q+\delta)$ and $f(\ldots, x_p+\delta, \mathrm{x}_q+\delta)$ in Theorem 2 are denoted by $f_{p,q}$, $f_{p+\delta,q}$, $f_{p,q+\delta}$ and $f_{p+\delta,q+\delta}$. The subcomponents that interact with variable

i are identified in line 7 by applying Theorem 2. It can be seen that the interactions of decision variables can be identified one by one in a single-pool detection manner. It is worth mentioning that the bold **G** in Algorithm 2 represents a set used to store the identified subcomponents. Each element in the set represents an individual subcomponent with a specific number of variables, which is denoted as G_i, where $i = 1, \ldots, m$.

Algorithm 2: $\mathbf{G} = \text{SPDG}(f, n)$

```
1  Construct the subcomponents: G={};
2  The first subcomponent G_1 is set to [1];
3  Calculate the f_{p,q};
4  for i = 2 → n do
5      Calculate the f_{p+δ,q} with respect to i;
6      ID^sub = [1, 2, ..., size(G)];
7      ID = INTERACT(f, i, G, ID^sub, f_{p,q}, f_{p+δ,q});
8      if ID is not empty then
9          Variable i and the interacted subcomponents
10         stored in ID are merged together;
11     else
12         Variable i is set to be a new subcomponent;
```

The detailed implementation of interaction identification is presented as Algorithm 3. This process is executed in a recursive manner [35]. Line 7 in Algorithm 3 employs a threshold range $(e_{\text{inf}}, e_{\text{sup}})$ for interaction identification by taking computational round-off errors into account. The greatest lower bound e_{inf} and the least upper bound e_{sup} for the roundoff error are estimated as follows [34]:

$$e_{\text{inf}} = \gamma_2 \cdot \max\{(f_{p,q} + f_{p+\delta,q+\delta}), (f_{p+\delta,q} + f_{p,q+\delta})\} \tag{13}$$

$$e_{\text{sup}} = \gamma_{\sqrt{n}} \cdot \max\{f_{p,q}, f_{p+\delta,q}, f_{p,q+\delta}, f_{p+\delta,q+\delta}\} \tag{14}$$

Specifically, γ_p in Eqs. (13) and (14) is calculated as follows:

$$\gamma_p = \frac{p \cdot \mu_{\text{M}}}{1 - p \cdot \mu_{\text{M}}}, \tag{15}$$

where μ_{M} is called the unit roundoff, which is half of machine epsilon ($\mu_{\text{M}} = \epsilon_{\text{M}}/2$). In the case of a double precision floating-point number (64 bits), ϵ_{M} is $2^{-52} \approx 2.22 \times 10^{-16}$.

Algorithm 3: $ID = \text{INTERACT}(f, i, \mathbf{G}, ID^{sub})$

```
1  ID is set to be empty: ID = [ ];
2  Calculate the f_{p,q+δ} with respect to G{ID^sub};
3  Calculate the f_{p+δ,q+δ} with respect to G{ID^sub} ∪ i;
4  Calculate the fitness change: Δ_1 = f_{p+δ,q} − f_{p,q};
5  Calculate the fitness change: Δ_2 = f_{p+δ,q+δ} − f_{p,q+δ};
6  ρ = |Δ_1 − Δ_2|;
7  if |ρ − e_inf| > |ρ − e_sup| then
8      if ID^sub contains only one element then
9          ID = ID ∪ ID^sub;
10     else
11         Divide ID^sub into ID^sub1 and ID^sub2;
12         ID^1 = INTERACT(f, i, G, ID^sub1);
13         ID^2 = INTERACT(f, i, G, ID^sub2);
14         ID = ID^1 ∪ ID^2;
```

For the values in the range $(e_{\text{inf}}, e_{\text{sup}})$, DG2 resets the threshold by weighting the bounds. It is noteworthy that we cannot obtain the entire interaction structure matrix [34] under the new identification framework. For simplicity, the middle point of the interval between e_{inf} and e_{sup} is used to set the threshold. Therefore, the judging condition of nonseparability is as follows:

$$\rho > \frac{e_{\text{inf}} + e_{\text{sup}}}{2} \Leftrightarrow |\rho - e_{\text{inf}}| > |\rho - e_{\text{sup}}| \tag{16}$$

When the interactions of variable i have been identified successfully, the subcomponents in **G** are updated accordingly in lines 8 to 11 of Algorithm 2. The above process is executed sequentially until all of the decision variables are identified.

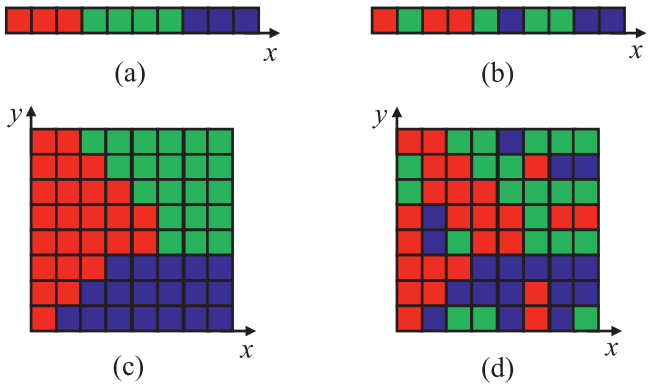


Fig. 3. Interaction structures of four separable problems with 1-dimensional or 2-dimensional topological information: (a) 1-dimensional case that satisfies the Proposal 1; (b) 1-dimensional case that does not satisfy the Proposal 1; (c) 2-dimensional case that satisfies the Proposal 1; (d) 2-dimensional case that does not satisfy the Proposal 1.

3.3. Topology-based decomposition

As discussed before, most of the existing decomposition methods neglect the associated topology information of decision variables in the interaction identification process. However, this information can be used to further improve the decomposition efficiency. Next, the technique of integrating the topology information into decomposition process is investigated in details.

The associated topology information of each decision variable x_i is denoted by t_i. In many real-world problems, this information can be 1-, 2- or 3-dimensional coordinates, which represent the geometric structure of the large-scale systems. In other words, t_i is represented by $\{x_i^t\}$, $\{x_i^t, y_i^t\}$ or $\{x_i^t, y_i^t, z_i^t\}$ for 1-, 2- or 3-dimensional physical problems respectively. In this study, a novel proposal about the topological consistency for subcomponents of large-scale problems is stated follows:

Proposal 1. *If the decision variables of an additively separable real-world optimization problem are accompanied with topology information, there is no topological overlapping among the subcomponents.*

It is noteworthy that the Proposal 1 is not a mathematical fact, but a general law obtained from the observations on various real-world problems. Among various real-world problems with positional variables, power system with positional modules [48], gene network with bunchy genetic blocks [49], and reservoir model with positional wells [50] are just a few examples that satisfy the topological consistency in Proposal 1. To elaborate, the interaction structures of four separable problems with 1-dimensional or 2-dimensional topological information are illustrated in Fig. 3. The given problems can be decomposed into three subcomponents which are marked by different colors.

Table 1
Containment test methods for the unidentified variables.

Dimensionality	Topological subcomponents	Test methods
1	Interval	Numerical comparison
2	Polygon	Ray casting [51]
3	Polyhedron	Determining triangle [52]

Each block represents an individual variable. We can see from Figs. 3(a) and 3(c) that the subcomponents of the problems are independent from a topological perspective. The subcomponents (i.e., intervals in 1-dimensional case or polygons in 2-dimensional case) do not interfere with each other under the constraint of topology structure. By contrast, the problems that do not satisfy the above topological consistency are illustrated in Figs. 3(b) and 3(d). The subcomponents interfere with each other topologically. As mentioned in Section 2, additive separability can represent the modular nature of many real-world optimization problems. This modular nature is always controlled by the topology structure of the given large-scale real-world problem. Thus, an efficient decomposition method for tackling the problems that satisfy the topological consistency (i.e., Proposal 1) is proposed in this section. Before we move on, two preconditions for applying the proposed topology-based decomposition method are listed as follows:

- The decision variables of the given separable problem are accompanied with topological information.
- The subcomponents of the given separable problem do not interfere with each other topologically.

In this study, the subcomponent with positional information is defined as topological subcomponent. With the Proposal 1, the decomposition task is to efficiently detect the topological subcomponents with the help of the associated topological information. Firstly, the priority of the variables for interaction identification is determined as follows:

$$i = \max_{j \in V} \min_{1 \le k \le m} \|x_j^t - s_k^t\|_2 \tag{17}$$

where V is a set that contains the index of unidentified variables, m represents the number of identified subcomponents, x_j^t denotes the location vector of jth variable and s_k^t denotes the central location vector of kth subcomponent.

An intuitive understanding of Eq. (17) is to search the variable that is farthest from the identified subcomponents. This can be achieved by maximizing the minimum distance between the variable and the identified subcomponents. After the interaction identification of each selected variable, the SPDF method merges the variable and the interacted subcomponent(s) and launches the identification for next variable. For the proposed topology-based method, an additional identification module for processing the unidentified variables that are topologically contained by the identified subcomponents is conducted. In this module, the contained unidentified variables can be directly removed from the variables' queue on 1-dimensional cases, while these variables are given priority to undergo the interaction identification with the associated subcomponents on multi-dimensional cases.

Algorithm 4 shows the implementation of the topology-based single-pool decomposition framework (TSPDF). We can see that this new framework can be easily obtained by integrating the variable selection strategy based on Eq. (17) and the additional identification module into the original SPDF. Firstly, the original sequential identification manner for variables (i.e., line 2 in Algorithm 1) is replaced by the new prioritized manner, as shown in lines 2 to 4 in Algorithm 4. Then, the unidentified variables that are topologically contained by the subcomponents are checked in line 9. The containment test methods of the unidentified variables for 1-D, 2-D and 3-D problems are listed in Table 1. Finally, the contained variables (i.e., V_c) are processed in line 10. The set of unidentified variables V and the set of identified subcomponents are updated accordingly. The above procedures are executed repeatedly until all the variables are identified (i.e., V is empty). An illustrative example for comparing the grouping process of SPDF and TSPDF in solving a separable problem with two subcomponents is shown in Fig. 4. The decision variables of the problem are accompanied with 1-dimensional coordinate information. We can see from Fig. 4(a) that SPDF sequentially identifies the interactions of the decision variables. The block with dotted boundary in each step represents an unidentified variable whose interaction relationship needs to be determined. There are totally seven steps are required to uncover the interaction structure of the given problem. Unlike SPDF, TSPDF utilizes the topological information to improve the grouping efficiency further. As shown in Fig. 4(b), the selected variable for interaction identification in step 1 is x_7, which is determined using Eq. (17). After the interaction identification for variable x_4 in step 2, the topologically contained variables x_5 and x_6 are removed from the set of unidentified variables and grouped into their associated subcomponent. Thus, the computational resources for identifying the interactions for x_5 and x_6 can be saved. The other subcomponent can be identified in the same manner, as shown in step 3. In short, if the preconditions of applying the Proposal 1 can be satisfied on a given problem, the grouping efficiency of SPDF can be greatly improved with the help of topological information, especially on the problems with high dimensionality. By imitating the conduction of the SPDG based on the SPDF, a topology-based single-pool differential grouping (TSPDG) method can be easily implemented.

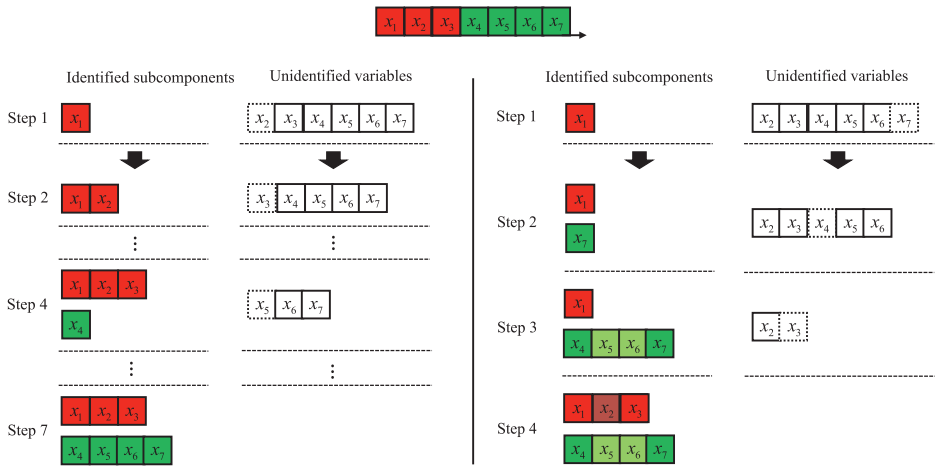
Algorithm 4: *Subcomponents* = TSPDF(f, n)

```
1  The first subcomponent G_1 is set to {1};
2  The set V of unidentified variables is set to {2, ..., n};
3  while V is not empty do
4      Select the variable i from the set V by applying equation
       Eq. (17);
5      if There are interactions between variable i and the
       subcomponents then
6          Variable i and the interacted subcomponents are
           merged together;
7      else
8          Variable i is set to be a new subcomponent;
9      Find the set of the unidentified variables V_c which are
       topologically contained by the identified
       subcomponents;
10     Determine the set of contained variables V_ci that
       interact with the associated subcomponents;
11     Update the set of unidentified variables: V = V − V_ci;
12     Update the subcomponents: G = G + V_ci;
```

4. Experimental methodology and results

4.1. Experimental methodology

In this section, the decomposition performance of the proposed SPDG and TSPDG are investigated in detail. Two metrics were selected to evaluate the performance of a decomposition method: (1) the number of function evaluations (FEs) used to decompose the problem; and (2) the grouping accuracy of decomposition algorithm.

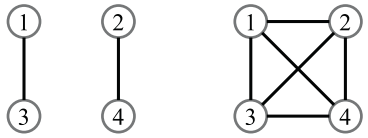


(a) SPDF (b) TSPDF

Fig. 4. An illustrative example for demonstrating the grouping process of SPDF and TSPDF in solving a separable problem with 1-dimensional topological information: (a) grouping process of SPDF; (b) grouping process of TSPDF.

Table 2
The parameter settings for all the decomposition methods used in the experiments.

Decomposition methods	Parameter Settings
FII	Threshold $\epsilon = 10^{-2}$
RDG	Control coefficient $\alpha = 10^{-12}$ and $k = 10$
DG2	Parameter-free
SPDG	Parameter-free
TSPDG	One-dimensional topological structure



(a) (b)

Fig. 5. Real interactions and identified interactions represented by (a) and (b) cannot be measured correctly by ρ method.

There are various metrics for grouping accuracy. FII considers whether the algorithm can get successful ideal decomposition. The metric employed by RDG may overstate the grouping accuracy of the decomposition methods. Even if several decomposition algorithms fail in getting an ideal decomposition, they can obtain 100% grouping accuracy under the metric. DG2 employs a metric proposed by Mei et al. [37] which consists of three measures: ρ_1 (interaction), ρ_2 (independence), and ρ_3 (interaction and independence).

However, the above ρ method is unable to get the right metric in some cases represented by the graphs shown in Fig. 5. For a function with two nonseparable subcomponents in Fig. 5(a), the identified result in Fig. 5(b) damages the accuracy of detecting interactions instead of independence. Under this circumstance,

Table 3
The experimental results of the proposed SPDG and the TSPDG decomposition methods when used to decompose the CEC'2010 benchmark problem. "DA" is the decomposition accuracy; "FEs" is the function evaluations used.

Benchmarks	Functions	FII($\epsilon = 10^{-2}$) DA/FEs	RDG($\alpha = 10^{-12}$) DA/FEs	DG2 DA/FEs	SPDG DA/FEs	TSPDG DA/FEs
CEC'2010	f_1	100%/3001	100%/3008	100%/500501	100%/2998	100%/2998
	f_2	100%/3001	100%/3008	100%/500501	100%/2998	100%/2998
	f_3	100%/3001	0.1%/6002	0.1%/500501	0.1%/2998	0.1%/4
	f_4	96.6%/3336	100%/4208	100%/500501	100%/4658	100%/3080
	f_5	100%/3051	100%/4154	100%/500501	100%/4722	100%/3080
	f_6	100%/3051	100%/50372	24.7%/500501	66.8%/14328	10.0%/2743
	f_7	100%/3051	100%/4232	100%/500501	100%/4601	100%/3080
	f_8	81.4%/3594	100%/5609	100%/500501	100%/4590	100%/4400
	f_9	100%/8010	100%/14036	100%/500501	100%/17563	100%/2217
	f_{10}	100%/8010	100%/14018	100%/500501	100%/18006	100%/2217
	f_{11}	99.8%/9616	75.1%/13694	75.1%/500501	75.1%/16100	75.1%/2416
	f_{12}	100%/8010	100%/14318	100%/500501	100%/17577	100%/2217
	f_{13}	100%/96183	100%/29243	100%/500501	100%/17819	100%/16421
	f_{14}	100%/23020	100%/20564	100%/500501	100%/18999	100%/1352
	f_{15}	100%/23572	100%/20522	100%/500501	100%/18955	100%/1352
	f_{16}	99.6%/30041	100%/20918	100%/500501	100%/18881	100%/1352
	f_{17}	100%/23020	100%/20768	100%/500501	100%/19005	100%/1352
	f_{18}	100%/369902	100%/49862	100%/500501	100%/27825	100%/26893
	f_{19}	100%/4001	100%/6002	100%/500501	100%/2998	100%/4
	f_{20}	100%/503500	100%/50876	100%/500501	100%/2998	100%/26411
	Mean	98.9%/56599	93.8%/17771	90.0%/500501	92.1%/11931	89.3%/5329

Table 4
The experimental results of the proposed SPDG and the TSPDG decomposition methods when used to decompose the CEC'2013 benchmark problem. "DA" is the decomposition accuracy; "FEs" is the function evaluations used.

Benchmarks	Functions	FII($\epsilon = 10^{-2}$) DA/FEs	RDG($\alpha = 10^{-12}$) DA/FEs	DG2 DA/FEs	SPDG DA/FEs	TSPDG DA/FEs
CEC'2013	f_1	100%/3001	100%/3008	100%/500501	100%/2998	100%/2998
	f_2	100%/3001	100%/3008	100%/500501	100%/2998	100%/2998
	f_3	100%/3001	0.1%/6005	0.1%/500501	0.1%/2998	0.1%/4
	f_4	59.7%/4817	100%/9842	100%/500501	100%/12324	100%/2988
	f_5	99.0%/10770	100%/10145	100%/500501	100%/12324	100%/2988
	f_6	99.4%/3645	54.5%/13238	51.1%/500501	51.1%/13725	51.1%/819
	f_7	91.1%/4654	86.4%/14492	97.5%/500501	100%/12397	100%/2988
	f_8	97.5%/18637	98.0%/19574	98.0%/500501	98.0%/23815	98.0%/3828
	f_9	100%/20676	100%/19343	100%/500501	100%/19075	100%/2959
	f_{10}	98.7%/17044	98.9%/19178	100%/500501	100%/18831	100%/2959
	f_{11}	7.3%/4687	6.9%/10319	100%/500501	100%/19071	100%/2959
	f_{12}	100%/503500	100%/50876	100%/500501	100%/2998	99.0%/1969
	f_{13}	62.8%/4123	99.8%/8315	100%/409966	98.2%/10880	100%/3227
	f_{14}	96.7%/4507	100%/16142	100%/409966	100%/7601	100%/3227
	f_{15}	100%/4003	100%/6125	100%/500501	100%/2998	100%/4
	Mean	87.5%/40673	83.0%/13974	89.8%/488430	89.8%/11002	89.9%/2461

the metric result provided by the ρ method is $\rho_1 = 100\%$, $\rho_2 = 0\%$ and $\rho_3 = 33.3\%$, which means the accuracy of detecting interactions is 100% and the accuracy of detecting independence is 0%. It is evident that ρ method fails to measure this function's grouping accuracy. Therefore, we propose a more robust metric for grouping accuracy in this section.

Two new measures are defined to indicate the accuracies of identifying the (1) losses of interactions; (2) surpluses of interactions. They are defined as follows:

$$\rho_l = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(\Theta^{ideal} \circ (\mathbf{1}_{n\times n} - \Theta))_{i,j}}{n^2} \times 100\% \tag{18}$$

$$\rho_s = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(\Theta \circ (\mathbf{1}_{n\times n} - \Theta^{ideal}))_{i,j}}{n^2} \times 100\% \tag{19}$$

where Θ^{ideal} is the ideal interaction structure. $\Theta_{i,j}^{ideal}$ equals 1 if variables i and j are interacted, and 0 otherwise. Θ is the interaction structure identified by the decomposition method. $\mathbf{1}_{n\times n}$ represents the interaction structure of fully nonseparable functions. An integrated measure can be established to indicate the overall grouping accuracy of decomposition method. It is defined as follows:

$$\mathrm{DA} = (1 - \rho_l - \rho_s) \times 100\% \tag{20}$$

We can see that the identified structure is identical to the ideal interaction structure only if DA = 100%. The performance of the proposed SPDG and the TSPDG methods was then compared to several other state-of-the-art decomposition algorithms, namely Differential Grouping 2 (DG2) [34], Fast Interdependency Identification (FII) [33] and Recursive Differential Grouping (RDG) [35]. The parameter settings for all the decomposition methods used in the experiments are shown in Table 2. Here, we assume that the subcomponents of the benchmark problems are topologically constrained by one-dimensional physical structure (i.e., the Proposal 1 can be utilized).

4.2. Comparative analysis of decomposition performance

Tables 3 and 4 list the decomposition results of the FII, RDG, DG2, SPDG and TSPDG methods on the CEC'2010 and CEC'2013 benchmark problems. In Table 3, "DA" represents the decomposition accuracy — if there are any losses or surpluses of variable interactions; FEs represents the decomposition efficiency — the number of function evaluations used in the decomposition stage. The detailed decomposition results for comparing the interaction-independence metric and the proposed metric can be found in the supplementary document accompanying this paper.[1]

4.2.1. Comparative analysis of grouping efficiency

Tables 3 and 4 contains the details for the grouping efficiency of FII, RDG, DG2, SPDG and TSPDG on the CEC'2010 and 2013 benchmark suites. As mentioned earlier, DG2 algorithm is capable of obtaining the entire interaction structure. However, the total number of functions evaluations required by DG2 is $(n^2+n+2)/2$. Except for DG2 method, we can see from Tables 3 and 4 that no single algorithm outperforms other algorithms with respect to the grouping efficiency. Moreover, it can be seen that the FII method is computationally intensive in decomposing the overlapping functions (e.g., f_{18} and f_{20} in the CEC'2010 benchmark). Thus, we compare the grouping efficiency of RDG, SPDG and TSPDG in the overall context. Fig. 6 shows the statistical results of FEs require by RDG, SPDG and TSPDG on the CEC'2010 and the CEC'2013 benchmark suites. Overall, RDG is comparable with the proposed SPDG in decomposing most of the benchmark problems. However, the outliers indicate that the RDG is less efficient in dealing with the overlapping functions. This shortcoming diminishes the practicability of the RDG algorithms. Furthermore, if the topology information of the LSGO problems can be integrated into the decomposition process, we can see that the proposed TSPDG method is capable of significantly improving the decomposition efficiency. Hence, by applying the proposed SPDG and TSPDG algorithms, the notorious cold-start problem of the static decomposition-based cooperative coevolution can be effectively alleviated.

4.2.2. Comparative analysis of grouping accuracy

Tables 3 and 4 list the overall decomposition accuracy (DA) of FII, RDG, DG2, SPDG, and TSPDG on the CEC'2010 and the CEC'2013 large-scale benchmark suites. Note that the ideal grouping is identified if DA is equal to 100%. We can see from Tables 3 and 4 that DG2, SPDG, and TSPDG algorithms show better generalizability over a wider range of functions than the other two decomposition algorithms. A presupposed control parameter may affect the decomposition accuracy of the FII and RDG methods, especially when there are nonuniform contribution of components in an objective function. It can be seen that the grouping

[1] Supplementary data can be found in Appendix A.

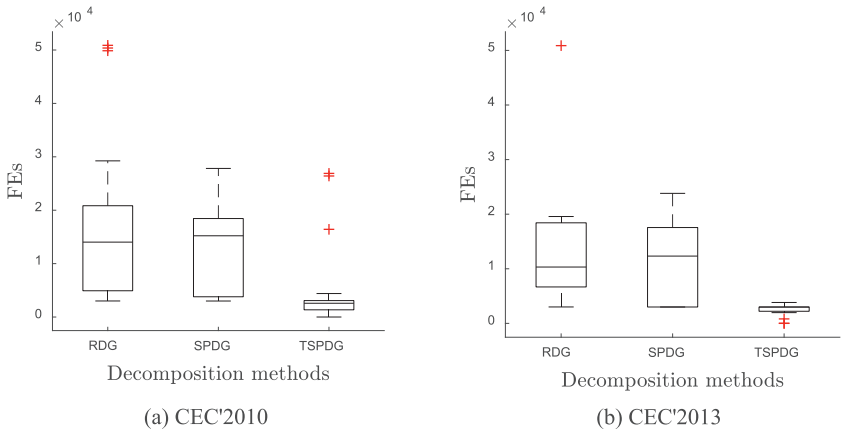


Fig. 6. The statistical results of FEs require by RDG, SPDG and TSPDG on the CEC'2010 and the CEC'2013 benchmark suites.

Table 5
Detailed comparison results of grouping accuracy on the CEC'2010 benchmark suite.

Fun.	FII			RDG			DG2			SPDG			TSPDG		
	$\epsilon = 10^{-2}$			$\alpha = 10^{-12}$			–			–			–		
	DA	A	I	DA	A	I	DA	A	I	DA	A	I	DA	A	I
f_3	100	×	✓	0.1	✓	×	0.1	✓	×	0.1	✓	×	0.1	✓	×
f_4	96.6	×	×	100	✓	✓	100	✓	✓	100	✓	✓	100	✓	✓
f_6	100	×	✓	100	×	✓	24.7	×	×	66.8	×	×	10.0	×	×
f_8	81.4	×	×	100	✓	✓	100	✓	✓	100	✓	✓	100	✓	✓
f_{11}	99.8	×	×	75.1	✓	×	75.1	✓	×	75.1	✓	×	75.1	✓	×
f_{16}	99.6	×	×	100	✓	✓	100	✓	✓	100	✓	✓	100	✓	✓
Success rate		0	2		5	4		5	3		5	3		5	3

accuracy of FII and RDG significantly decrease in CEC'2013 benchmark suite compared with CEC'2010, and this is precisely due to the imbalance property in CEC'2013 benchmark suite [36]. Next, we further investigated the grouping accuracy of the five decomposition algorithms in detail.

The detailed decomposition accuracies on the CEC'2010 benchmark functions are listed in Table 5 except that all the above five algorithms achieve ideal decomposition. We use I to indicate whether the algorithm obtains a successful ideal decomposition which satisfies Definition 1 (i.e., general separability), and we use A to indicate whether the algorithm gets a successful additively decomposition which satisfies Definition 2 (i.e., additive separability). In particular, here we take f_3 from the CEC'2010 benchmark as an example. Theoretically, f_3 is a fully separable problem from a perspective of the general separability, but a fully nonseparable problem from a perspective of the additive separability. As can be observed from Table 5, FII obtains an ideal decomposition on f_3 from a perspective of the general separability (i.e., I is checked), while the rest of the methods get successful decomposition from a perspective of the additive separability (i.e., A is checked) although their decomposition accuracies are very low. Tables 3 and 4 show that the overall accuracies of DG2, SPDG and TSPDG appear to be lower than FII and RDG. It is due to the three instances of the Ackley function (i.e., f_3, f_6 and f_{11} in the CEC'2010 benchmark) which affects the mean values of DA in Tables 3 and 4. It should be noted that the Ackley function is not *additively separable* [53]. Theoretically, all five of these algorithms are incapable of obtaining an ideal decomposition for such function. FII and RDG's successful decomposition on these functions is attributed to a 'suitable' control parameter. A larger ϵ makes FII successfully identify the separable variables that satisfy Definition 1 rather than Definition 2. However, it is risky to adopt a large ϵ in practice.

Table 6 shows the detailed comparison results of grouping accuracy on the CEC'2013 benchmark suite. We can see that the FII and RDG methods perform poorly on this benchmark suite. This is because the presupposed parameter ϵ of FII or global parameter α of RDG is not suitable for imbalanced functions. Note that the DG2, SPDG and TSPDG algorithms show superior performance in decomposing the additively separable functions. However, these methods are incapable of decomposing the generally separable functions. Table 6 clearly shows that the grouping accuracy of DG2, SPDG and TSPDG are very low on f_3 and f_6 in the CEC'2013 benchmark. It is notable that these two functions are generally separable (i.e., Ackley function). We believe that as a general and efficient decomposition framework, SPDF can be used with a generalized interaction identification criterion, to discover the separability of the generally separable problems. This will be the subject of our future work.

4.3. Comparative analysis of optimization performance

In this section, firstly we compare the optimization performance of FII, RDG, DG2, SPDG, and TSPDG methods when embedded in a sophisticated cooperative coevolutionary framework. In order to obtain a more comprehensive comparison, the computational budget is divided into two cases: limited computational resources and sufficient computational resources. The empirical results are based on the CEC'2010 and the CEC'2013 benchmark suites [30]. Then, we show that in conjunction with a sophisticated cooperative coevolutionary framework, the proposed decomposition methods show competitive results to several state-of-the-art LSGO algorithms.

4.3.1. Comparison under limited computational resources

Generally, large-scale real-world problems are time-consuming due to their complex features. It is common that the computational resources for these problems are limited. The application of cooperative coevolution in solving such problems is limited by the efficiency of decomposition method. The decomposition

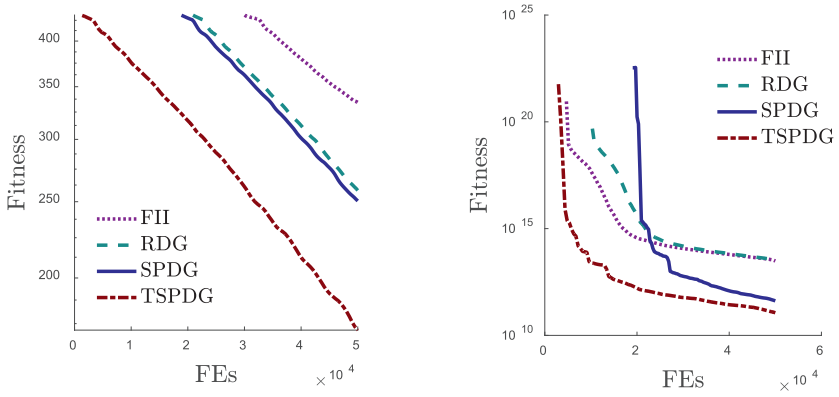
Table 6
Detailed comparison results of grouping accuracy on the CEC'2013 benchmark suite.

Fun.	FII			RDG			DG2			SPDG			TSPDG		
	$\epsilon = 10^{-2}$			$\alpha = 10^{-12}$			–			–			–		
	DA	A	I	DA	A	I	DA	A	I	DA	A	I	DA	A	I
f_3	100	×	✓	0.1	✓	×	0.1	✓	×	0.1	✓	×	0.1	✓	×
f_4	59.7	×	×	100	✓	✓	100	✓	✓	100	✓	✓	100	✓	✓
f_5	99.0	×	×	100	✓	✓	100	✓	✓	100	✓	✓	100	✓	✓
f_6	99.4	×	×	54.5	×	×	51.1	✓	×	51.1	✓	×	51.1	✓	×
f_7	91.1	×	×	86.4	×	×	97.5	×	×	100	✓	✓	100	✓	✓
f_8	97.5	×	×	98.0	×	×	98.0	×	×	98.0	×	×	98.0	×	×
f_{10}	98.7	×	×	98.9	×	×	100	✓	✓	100	✓	✓	100	✓	✓
f_{11}	7.3	×	×	6.9	×	×	100	✓	✓	100	✓	✓	100	✓	✓
Success rate		0	1		3	2		6	4		7	5		7	5

Table 7
Detailed optimization performance comparison of FII, RDG, DG2, SPDG, and TSPDG on the CEC'2010 and CEC'2013 benchmarks under limited computational resources. The symbol '↑', '↓', and '≈' indicate that the baseline (TSPDG) performs better than, worse than, and statistically similar to the algorithm under consideration.

		CEC'2010					CEC'2013				
		FII	RDG	DG2	SPDG	TSPDG	FII	RDG	DG2	SPDG	TSPDG
f_1	Mean	6.20e+08$^{\approx}$	5.94e+08$^{\approx}$	9.09e+11$^{\uparrow}$	7.81e+08$^{\approx}$	6.35e+08	6.87e+08$^{\approx}$	7.17e+08$^{\approx}$	8.93e+11$^{\uparrow}$	6.11e+08$^{\approx}$	7.63e+08
	StDev	2.73e+08	1.70e+08	–	5.64e+08	2.79e+08	5.21e+08	6.05e+08	–	1.83e+08	3.47e+08
f_2	Mean	4.31e+03$^{\approx}$	4.37e+03$^{\approx}$	4.25e+04$^{\uparrow}$	4.37e+03$^{\approx}$	4.32e+03	5.36e+03$^{\approx}$	5.32e+03$^{\approx}$	1.28e+05$^{\uparrow}$	5.33e+03$^{\approx}$	5.32e+03
	StDev	1.17e+02	1.43e+02	–	1.19e+02	1.66e+02	2.16e+02	2.35e+02	–	1.47e+01	2.84e+02
f_3	Mean	1.86e+01$^{\downarrow}$	2.17e+01$^{\approx}$	2.17e+01$^{\approx}$	2.17e+01$^{\approx}$	2.17e+01	2.06e+01$^{\downarrow}$	2.17e+01$^{\approx}$	2.17e+01$^{\approx}$	2.17e+01$^{\approx}$	2.17e+01
	StDev	1.97e−01	1.10e−02	–	1.24e−02	1.24e−02	1.89e−02	9.90e−03	–	1.51e−02	8.80e−03
f_4	Mean	6.71e+11$^{\downarrow}$	1.12e+12$^{\approx}$	3.51e+16$^{\uparrow}$	1.19e+12$^{\approx}$	1.03e+12	4.85e+13$^{\uparrow}$	2.87e+10$^{\uparrow}$	2.75e+14$^{\uparrow}$	3.95e+10$^{\uparrow}$	1.58e+10
	StDev	2.69e+11	5.47e+11	–	5.03e+11	3.58e+11	1.33e+13	1.06e+10	–	1.70e+10	4.84e+09
f_5	Mean	9.65e+07$^{\approx}$	1.01e+08$^{\approx}$	1.82e+09$^{\uparrow}$	9.61e+07$^{\approx}$	1.12e+08	2.29e+06$^{\approx}$	2.46e+06$^{\approx}$	4.91e+08$^{\uparrow}$	2.30e+06$^{\approx}$	2.32e+06
	StDev	2.59e+07	1.90e+07	–	1.35e+07	3.38e+07	3.42e+05	3.69e+05	–	5.51e+05	4.04e+05
f_6	Mean	2.15e+06$^{\downarrow}$	2.08e+07$^{\uparrow}$	2.13e+07$^{\uparrow}$	4.24e+06$^{\uparrow}$	2.16e+06	1.08e+06$^{\approx}$	1.07e+06$^{\approx}$	1.07e+06$^{\approx}$	1.07e+06$^{\approx}$	1.07e+06
	StDev	6.60e+06	–	–	8.70e+06	6.63e+06	1.94e+03	1.73e+03	–	2.69e+03	1.44e+03
f_7	Mean	1.79e+05$^{\approx}$	1.98e+05$^{\approx}$	2.57e+14$^{\uparrow}$	1.97e+05$^{\approx}$	1.84e+05	1.22e+11$^{\uparrow}$	1.82e+11$^{\uparrow}$	1.79e+19$^{\uparrow}$	8.46e+09$^{\uparrow}$	3.87e+09
	StDev	2.47e+04	2.78e+04	–	3.21e+04	1.75e+04	7.82e+10	4.99e+10	–	2.42e+09	1.61e+09
f_8	Mean	7.42e+07$^{\approx}$	7.76e+07$^{\approx}$	1.21e+18$^{\uparrow}$	5.88e+07$^{\approx}$	1.10e+08	6.85e+12$^{\uparrow}$	1.30e+13$^{\uparrow}$	1.94e+19$^{\uparrow}$	2.77e+13$^{\uparrow}$	8.81e+11
	StDev	1.18e+08	1.40e+08	–	5.39e+07	1.76e+08	2.99e+12	1.17e+13	–	2.26e+13	5.76e+11
f_9	Mean	5.92e+09$^{\uparrow}$	8.10e+09$^{\uparrow}$	9.90e+11$^{\uparrow}$	1.09e+10$^{\uparrow}$	4.17e+09	1.91e+08$^{\uparrow}$	1.60e+08$^{\approx}$	2.42e+10$^{\uparrow}$	1.86e+08$^{\uparrow}$	1.62e+08
	StDev	8.51e+08	1.26e+09	–	2.76e+09	5.33e+08	3.17e+07	3.61e+07	–	3.52e+07	3.69e+07
f_{10}	Mean	7.26e+03$^{\uparrow}$	7.76e+03$^{\uparrow}$	4.06e+04$^{\uparrow}$	8.07e+03$^{\uparrow}$	6.75e+03	9.62e+07$^{\approx}$	9.71e+07$^{\approx}$	9.63e+07$^{\approx}$	9.72e+07$^{\uparrow}$	9.60e+07
	StDev	4.16e+02	2.46e+02	–	3.05e+02	3.39e+02	1.43e+06	5.04e+05	–	5.23e+05	1.79e+06
f_{11}	Mean	1.38e+02$^{\uparrow}$	8.89e+01$^{\approx}$	2.38e+02$^{\uparrow}$	9.24e+01$^{\uparrow}$	7.05e+01	3.40e+13$^{\uparrow}$	4.08e+13$^{\uparrow}$	1.84e+22$^{\uparrow}$	4.23e+11$^{\uparrow}$	1.16e+11
	StDev	2.02e+01	3.28e+01	–	2.72e+01	3.18e+02	2.78e+13	2.60e+13	–	1.98e+11	9.33e+10
f_{12}	Mean	2.28e+06$^{\uparrow}$	2.88e+06$^{\uparrow}$	4.27e+09$^{\uparrow}$	3.55e+06$^{\uparrow}$	1.87e+06	9.72e+12$^{\uparrow}$	1.70e+12$^{\uparrow}$	3.00e+13$^{\uparrow}$	1.57e+06$^{\approx}$	1.54e+06
	StDev	2.56e+05	3.72e+05	–	5.13e+05	1.58e+05	–	–	–	3.40e+06	1.90e+06
f_{13}	Mean	4.11e+12$^{\uparrow}$	1.82e+09$^{\uparrow}$	1.35e+13$^{\uparrow}$	6.90e+06$^{\uparrow}$	4.65e+06	1.11e+13$^{\downarrow}$	1.58e+13$^{\approx}$	8.11e+20$^{\uparrow}$	1.88e+13$^{\approx}$	1.60e+13
	StDev	–	4.27e+08	–	3.50e+06	2.47e+06	3.63e+12	6.14e+12	–	6.89e+12	4.03e+12
f_{14}	Mean	2.76e+10$^{\uparrow}$	2.64e+10$^{\uparrow}$	8.34e+11$^{\uparrow}$	2.28e+10$^{\uparrow}$	9.77e+09	9.49e+13$^{\approx}$	1.21e+14$^{\uparrow}$	1.66e+21$^{\uparrow}$	8.85e+13$^{\approx}$	9.37e+13
	StDev	1.91e+09	2.62e+09	–	2.22e+09	8.74e+08	4.93e+13	3.93e+13	–	3.62e+13	4.36e+13
f_{15}	Mean	1.12e+04$^{\uparrow}$	1.10e+04$^{\uparrow}$	4.25e+04$^{\uparrow}$	1.10e+04$^{\uparrow}$	9.75e+03	9.82e+10$^{\approx}$	9.90e+10$^{\approx}$	1.19e+12$^{\uparrow}$	9.38e+10$^{\approx}$	8.70e+10
	StDev	1.60e+02	1.72e+02	–	4.03e+02	4.81e+02	3.63e+10	3.18e+10	–	2.98e+10	2.15e+10
f_{16}	Mean	3.34e+02$^{\uparrow}$	2.60e+02$^{\uparrow}$	4.33e+02$^{\uparrow}$	2.49e+02$^{\uparrow}$	1.67e+02					
	StDev	8.98e+00	7.15e+00	–	1.15e+01	2.07e+01					
f_{17}	Mean	1.02e+07$^{\uparrow}$	9.74e+06$^{\uparrow}$	9.25e+09$^{\uparrow}$	9.02e+06$^{\uparrow}$	6.21e+06					
	StDev	8.99e+05	6.20e+05	–	4.91e+05	6.08e+05					
f_{18}	Mean	8.86e+12$^{\uparrow}$	8.03e+12$^{\uparrow}$	2.83e+13$^{\uparrow}$	2.99e+09$^{\uparrow}$	2.09e+09					
	StDev	–	4.80e+11	–	4.28e+08	2.68e+08					
f_{19}	Mean	7.89e+08$^{\approx}$	8.18e+08$^{\approx}$	3.52e+12$^{\uparrow}$	8.50e+08$^{\approx}$	8.27e+08					
	StDev	7.49e+07	9.28e+07	–	9.14e+07	1.23e+08					
f_{20}	Mean	1.06e+13$^{\uparrow}$	1.66e+12$^{\uparrow}$	3.13e+13$^{\uparrow}$	3.90e+06$^{\downarrow}$	3.66e+09					
	StDev	–	–	–	6.57e+06	5.65e+08					
	w/t/l	3/6/11	0/9/11	0/1/19	1/8/11	–	2/7/6	0/9/6	0/3/12	0/9/6	–

method must be selected carefully. Thus, we investigate the performance of the above five decomposition algorithms in solving the CEC'1010 and the CEC'2013 benchmarks under limited computational budgets. The maximum number of fitness evaluations is set to 5×10^4. Table 7 contains the detailed experimental results to compare the performance of FII, RDG, DG2, SPDG, and TSPDG within a co-evolutionary framework. We employ a sophisticated CC framework (CCFR) in which subcomponents with



(a) f_{16} in CEC'2010 (b) f_{11} in CEC'2013

Fig. 7. The convergence curves of FII, RDG, SPDG, and TSPDG methods when embedded into the CCFR framework with the CMA-ES optimizer to solve two instances from the CEC'2010 and the CEC'2013. The horizontal axis represents the number of FEs used in the evolutionary process. The vertical axis represents the median of the best fitness found.

Table 8
Performance comparison of FII, RDG, DG2, and SPDG against TSPDG on the CEC'2010 and CEC'2013 benchmarks under sufficient computational resources. The first four methods' numbers of wins, ties, and losses against the TSPDG are reported.

	CEC'2010				CEC'2013			
	FII	RDG	DG2	SPDG	FII	RDG	DG2	SPDG
w/t/l	2/6/12	0/10/10	0/1/19	1/8/11	1/7/7	0/8/7	0/1/14	0/9/6

higher contribution to the overall solution quality are given more computational resources [54]. In this study, a popular and competitive component optimizer named CMAES [55] is employed for CCFR. The number of iterations for each subcomponent is set to 5. The parameter settings for CMA-ES are consistent with the original reports. The number of objective function evaluations used in the decomposition stage is deducted from the maximum available evaluations. All experimental results are based on 25 independent runs. To test the statistical significance of the results, all other decomposition methods are compared with the baseline (TSPDG) using a two-tailed Wilcoxon rank-sum test with $\alpha = 0.05$.

Performance comparison of FII, RDG, DG2, and SPDG against TSPDG is listed in the last line of Table 7. The first four methods' numbers of wins, ties, and losses against the TSPDG are reported. A dash symbol (–) indicates that there are no available computational resources after the problem decomposition. It is clear that the FII, RDG and DG2 always meet the above situation. Their optimal fitness values recorded during the decomposition phase are reported as the mean in Table 7 (e.g., RDG on f_6 from the CEC'2010). This drawback limits the application of these three methods in solving problems under limited computational budgets. Fortunately, the proposed SPDF-based methods work well in this case. Moreover, by incorporating the topology information, we can see that the TSPDG outperforms RDG, DG2 and SPDG in solving the two benchmark problems by a wide margin, except for f_{20} in CEC'2010 of SPDG. This exception can be attributed to the complete overlapping structure of f_{20}. Besides, it can be seen that the TSPDG performs worse than the FII in a small number of instances, including the overlapping functions and the generally separable functions. However, the win of the FII should be attributed to a prespecified but suitable control parameter (i.e., $\epsilon = 10^{-2}$). In short, when SPDF-based decomposition methods are embedded in a CC framework, the optimization results are better than results from the three state-of-the-art decomposition methods. In particular, if the topology information can be utilized, the optimization performance can be further boosted by TSPDG. Convergence curves of two typical optimization instances from the CEC'2010 and the CEC'2013 benchmarks are shown in Fig. 7.

4.3.2. Comparison under sufficient computational resources

In this part, the optimization performance of the five decomposition methods are investigated under sufficient computational resources. The maximum number of fitness evaluations is set to 3×10^6 as suggested by Li et al. [30]. The prespecified number of the evolutionary generations for CMA-ES is set to 100. Similarly, the number of objective function evaluations used in the decomposition stage is deducted from the maximum available evaluations. All experimental results are based on 25 independent runs. To test the statistical significance of the results, all other decomposition methods are compared with the baseline (TSPDG) using a two-tailed Wilcoxon rank-sum test with $\alpha = 0.05$.

For the sake of brevity, an overall optimization performance comparison is shown in Table 8. We can see that the TSPDG shows superior performance as compared to other methods. The results verified that an efficient and accurate decomposition method is essential in the application of the cooperative coevolution.

4.3.3. Comparison with the state-of-the-art

Finally, we compare the performance of the CCFR that employs SPDF-based methods as its decomposition methods with two well-known algorithms: Multiple Offspring Framework (MOS) [56] and MA-SW-Chains [57]. The parameter settings of these two algorithms match the reported values in the original papers. MA-SW-Chains obtained the best overall results in the CEC'2010 competition, and MOS was the best performing algorithm in the CEC'2013 competition on the LSGO. Table 9 contains the experimental results using 25 independent runs on f_1–f_{18} from the CEC'2010 and f_1–f_{11} from the CEC'2013 large-scale benchmark suites. We mainly focused on the separable functions, and this is because no decomposition is done for f_{19}–f_{20} from the CEC'2010 and f_{12}–f_{15} from the CEC'2013, in which case CCFR-CMAES degrades to CMAES.

Table 9 shows that no single algorithm outperforms other algorithms. Overall, CCFR-CMAES assisted by TSPDG shows more superior performance as compared to other algorithms, especially on the partially separable functions. It has been shown that a cooperative co-evolutionary framework can scale up the performance of many optimizers [19,22,37,58]. We believe that as an efficient decomposition method with high accuracy, SPDF-based methods can be used with other promising LSGO algorithms such as MOS and MA-SW-Chains to improve their optimization performance further.

Table 9
Experimental results of CCFR-SPDG, CCFR-TSPDG, MOS, and MA-SW-Chains on the CEC'2010 and the CEC'2013 large-scale benchmark suites using 25 independent runs. The highlighted entries are significantly better (Wilcoxon rank-sum test with Holm p-value correction, $\alpha = 0.05$).

		CEC'2010				CEC'2013			
		CCFR-SPDG	CCFR-TSPDG	MOS	MA-SW-Chains	CCFR-SPDG	CCFR-TSPDG	MOS	MA-SW-Chains
f_1	Mean	**0.00e+00**	**0.00e+00**	**0.00e+00**	3.80e−14	**0.00e+00**	**0.00e+00**	**0.00e+00**	1.14e−12
	StDev	0.00e+00	0.00e+00	0.00e+00	4.91e−14	5.21e+08	6.05e+08	0.00e+00	1.28e−12
f_2	Mean	1.07e+03	1.10e+03	**0.00e+00**	8.40e+02	9.64e+02	9.89e+02	**8.23e+02**	1.18e+03
	StDev	3.85e+01	2.65e+01	0.00e+00	4.88e+01	3.71e+01	2.86e+01	4.69e+01	1.84e+02
f_3	Mean	2.17e+01	2.17e+01	1.65e−12	**5.76e−13**	2.17e+01	2.17e+01	1.69e−12	**6.78e−13**
	StDev	8.90e−03	1.14e−02	6.44e−14	2.73e−13	1.29e−02	7.50e−03	9.16e−14	2.28e−13
f_4	Mean	1.83e+00	**3.52e−01**	1.70e+10	2.97e+11	8.22e−04	**2.54e−04**	8.73e+07	3.80e+09
	StDev	2.73e+00	3.76e−01	6.39e+09	6.19e+10	1.20e−03	3.86e−04	3.11e+07	2.70e+09
f_5	Mean	**9.13e+07**	**9.14e+07**	1.07e+08	2.18e+08	**2.29e+06**	**2.14e+06**	6.89e+06	**2.26e+06**
	StDev	1.16e+07	1.43e+07	1.35e+07	3.38e+07	1.98e+05	1.46e+05	9.16e+05	1.36e+06
f_6	Mean	2.16e+01	2.17e+01	**1.11e−07**	1.42e+05	1.06e+06	1.06e+06	**1.43e+05**	1.07e+04
	StDev	4.23e−02	8.80e−03	5.88e−08	3.95e+05	1.80e+04	2.32e+04	6.86e+04	2.09e+04
f_7	Mean	9.45e−06	2.03e−08	**0.00e+00**	1.17e+02	**1.96e+03**	4.41e+04	4.65e+03	3.78e+06
	StDev	2.31e−05	7.94e−08	0.00e+00	2.37e+02	2.85e+03	7.07e+04	1.06e+04	8.46e+05
f_8	Mean	4.37e+02	4.55e+02	**1.40e+00**	6.90e+06	**1.45e+04**	**1.55e+04**	2.85e+12	4.63e+13
	StDev	1.40e+02	1.28e+02	7.01e+00	1.90e+07	4.37e+03	5.95e+03	1.44e+12	9.18e+12
f_9	Mean	5.66e−05	**4.23e−05**	3.59e+06	1.49e+07	1.52e+08	1.60e+08	3.99e+08	**1.14e+08**
	StDev	1.73e−05	1.53e−05	4.89e+05	1.61e+06	2.54e+07	1.48e+07	6.26e+07	2.05e+07
f_{10}	Mean	**1.52e+03**	**1.54e+03**	3.81e+03	2.01e+03	9.62e+07	9.71e+07	9.38e+05	**3.66e+04**
	StDev	4.45e+01	5.24e+01	1.62e+02	1.59e+02	1.43e+06	5.04e+05	4.79e+05	6.17e+04
f_{11}	Mean	7.50e+01	6.36e+01	1.91e+02	**3.86e+01**	1.75e+04	**1.39e+02**	1.73e+07	2.10e+08
	StDev	1.90e+01	1.72e+01	4.01e−01	8.06e+00	7.76e+04	9.74e+01	5.04e+06	2.43e+07
f_{12}	Mean	9.36e−12	2.95e−11	**0.00e+00**	3.24e−06				
	StDev	9.80e−12	5.50e−11	0.00e+00	5.78e−07				
f_{13}	Mean	3.99e+01	**2.59e+00**	8.23e+02	9.83e+02				
	StDev	2.59e+01	1.95e+00	6.77e+02	5.66e+02				
f_{14}	Mean	**0.00e+00**	**0.00e+00**	9.69e+06	3.25e+07				
	StDev	0.00e+00	0.00e+00	6.71e+05	2.46e+06				
f_{15}	Mean	**1.98e+03**	**2.00e+03**	7.44e+03	2.68e+03				
	StDev	5.96e+01	6.61e+01	1.90e+02	9.95e+01				
f_{16}	Mean	**6.82e+01**	**7.25e+01**	3.79e+02	9.95e+01				
	StDev	2.90e+01	2.70e+01	1.83e+01	1.53e+01				
f_{17}	Mean	**0.00e+00**	**0.00e+00**	2.73e−07	1.27e+00				
	StDev	0.00e+00	0.00e+00	7.76e−08	1.24e−01				
f_{18}	Mean	**6.18e+00**	**6.33e+00**	1.77e+03	1.57e+03				
	StDev	3.54e+00	2.96e+00	9.57e+02	6.73e+02				

4.4. Application on production optimization problem

In this section, we demonstrate the efficacy of TSPDG against other state-of-the-art decomposition methods on a real-world finite difference simulation-based optimization case study from the petroleum industry. As is well known, such simulations are often computationally expensive. A single run for such simulations can take from several minutes to several hours, thereby limiting the number of simulations (i.e., FEs) for optimizing such problems.

Before we proceed to the details of our model representation and experimental settings, we present a brief overview of the production optimization in the *closed-loop reservoir management* (CLRM), which mainly consists of model calibration and model optimization. In the field of petroleum engineering, these two parts are termed history matching and production optimization respectively. For full details on CLRM and history matching, the reader is referred to [59,60]. Production optimization aims to increase ultimate oil recovery substantially by developing an improved operating plan for a particular reservoir of interest. Generally, this operating plan is referred to the well controls such as production rates, injection rates, and bottom hole pressure, etc.

For the production optimization problem considered here, we wish to obtain the optimal well controls that maximize the *net present value* (NPV) of production during the time period corresponding to the expected life of the reservoir. For a fixed reservoir model m and decision vector of well controls u, NPV is defined as follows [61]:

$$J(u, m, y) = \sum_{n=1}^{L} \left[\sum_{j=1}^{N_p} \left(r_o q_{o,j}^n - r_w q_{w,j}^n \right) - \sum_{j=1}^{N_I} r_{wi} q_{wi,j}^n \right] \tag{21}$$

where J is a function of the decision vector u, the given reservoir model m and the system-state vector y, which includes all the primary variables of the reservoir simulator. The vector u represents the aforementioned operating plan which needs to be optimized. In Eq. (21), L is the total number of control timesteps and N_p and N_I, respectively, denote the total number of producing wells and the total number of injecting wells. The scalars r_o, r_w, and r_{wi} represent, respectively, the oil revenue, the cost of handling produced water, and the cost of water-injection. $q_{o,j}^n$ and $q_{w,j}^n$ are, respectively, the average oil- and water-production rate of the jth producer during the nth control timestep; $q_{wi,i}^n$ is the average water-injection rate of the ith water-injection well during the nth control timestep.

In this demonstration, we consider the production optimization of a large-scale reservoir which contains 100 injecting wells and 225 producing wells. Fig. 8 shows the permeability distribution and the well locations of the given reservoir. It is noted that this large-scale reservoir consists of 25 sub-regions and each of

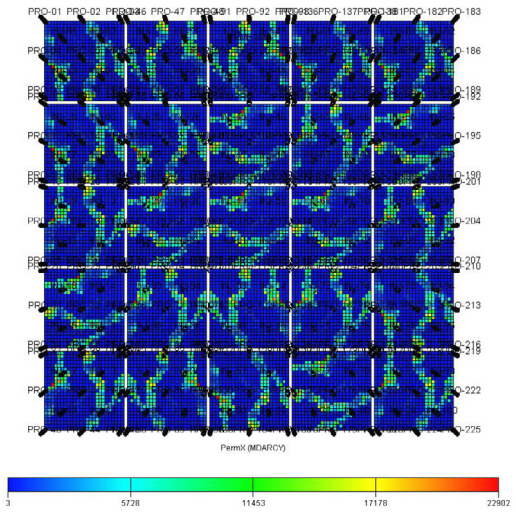


Fig. 8. Permeability distribution and well locations of the given reservoir model.

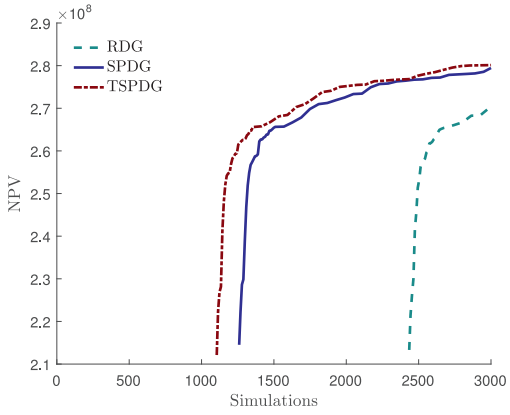


Fig. 9. The convergence curves of RDG, SPDG, and TSPDG methods when embedded into the CCFR framework with the SaNSDE optimizer to solve the given production optimization problem.

these regions contains 4 injecting wells and 9 producing wells. Therefore, the original large-scale problem is additively separable and can be efficiently solved with the help of cooperative coevolution.

For the numerical experiments, the number of control timesteps is set to 1 and the duration is set to 5 years. The controls for all 325 wells are flow rates (i.e., production rates or injection rates). Thus, the total number of control variables in this example is 325. The lower and upper bounds of the control variables are set to be 0 and 200, respectively. So far, a box-constrained large-scale expensive optimization problem is conducted. The parameter settings used for the five decomposition methods are identical to those considered for the CEC benchmark suites. In particular, for TSPDG method, the topology information here is two-dimensional coordinate and the containment test method is *ray casting* [51]. The CCFR framework is employed and the subcomponent optimizer is set to SaNSDE [62]. The population size and the number of iterations for each subcomponent is set to be 10 and 5, respectively. The maximum number of simulations is set to 3000. as the simulation is time-consuming. The number of simulations used by the decomposition methods is deducted from the maximum available simulations in the optimization stage. All experimental results are based on 5 independent runs.

As is revealed in Fig. 9, the proposed methods show superior performance with respect to grouping efficiency as compared to other state-of-the-art decomposition methods. The convergence curves of FII and DG2 methods are missing due to their low decomposition efficiency. In other words, there are no available computational resources for FII and DG2 in the optimization stage. The numbers of simulations used by FII and DG2 are 8150 and 105954, respectively. Moreover, it can be seen that the TSPDG consumes less simulations than the SPDG. The decomposition efficiency is further improved with the help of topology information, therefore the notorious cold-start problem is alleviated further. This means a lot for application of the cooperative coevolution on real-world problems.

5. Conclusion

In this paper, we firstly revisited the theorem of interaction identification and derived the theorem in a simpler way. Then, we proposed an efficient single-pool decomposition framework — SPDF, which can decompose an n-dimensional problem efficiently. In the identification process, the interactions of decision variables are identified in an ordinal fashion. Moreover, a topology-based decomposition strategy was proposed in this paper. By incorporating the topology information into the decomposition process, the grouping efficiency can be improved further. This approach can promote the application of static decomposition-based cooperative coevolution due to the alleviation of notorious cold-start problem. To evaluate the effectiveness of the proposed methods, comprehensive studies on the CEC large-scale benchmarks and a real-world application from the production optimization problem are conducted. Significantly, when SPDF-based methods were embedded into a CC framework, the optimization results were better than results from three other state-of-the-art decomposition methods.

In a future study, we plan to focus on the modification of the SPDF-based decomposition methods to the multi-objective optimization and the expensive optimization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Xiaoming Xue: Conceptualization, Methodology, Software, Data curation, Writing - original draft, Writing - review & editing. **Kai Zhang:** Methodology, Investigation, Resources, Funding acquisition, Supervision. **Rupeng Li:** Methodology, Writing - original draft, Validation. **Liming Zhang:** Visualization, Resources, Supervision. **Chuanjin Yao:** Project administration. **Jian Wang:** Visualization, Validation, Supervision. **Jun Yao:** Funding acquisition.

Acknowledgments

This work is supported by the "National Natural Science Foundation of China" under Grant 51722406, 51874335 and 51674280, the "Natural Science Foundation of Shandong Province, China" under Grant JQ201808 and ZR2019JQ21, the "National Major Science and Technology Projects of China" under Grant 2016ZX05025001-006.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.asoc.2020.106295.

References

[1] Q. Yang, W. Chen, T. Gu, H. Zhang, J.D. Deng, Y. Li, J. Zhang, Segment-based predominant learning swarm optimizer for large-scale optimization, IEEE Trans. Cybern. 47 (9) (2017) 2896–2910.

[2] M. Lozano, D. Molina, F. Herrera, Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems, Soft Comput. 15 (11) (2011) 2085–2087.

[3] M. Olhofer, Y. Jin, B. Sendhoff, Adaptive encoding for aerodynamic shape optimization using evolution strategies, in: Proceedings of the 2001 Congress on Evolutionary Computation, Vol. 1, IEEE, 2001, pp. 576–583.

[4] X. Li, K. Tang, P.N. Suganthan, Z. Yang, Editorial for the special issue of Information Sciences Journal (ISJ) on nature-inspired algorithms for large scale global optimization, Inform. Sci. 316 (C) (2015) 437–439.

[5] G.N. Vanderplaats, Very Large Scale Optimization, National Aeronautics and Space Administration, Langley Research Center, 2002.

[6] Z. Yang, B. Sendhoff, K. Tang, X. Yao, Target shape design optimization by evolving B-splines with cooperative coevolution, Appl. Soft Comput. 48 (2016) 672–682.

[7] C. Wang, J. Gao, High-dimensional waveform inversion with cooperative coevolutionary differential evolution algorithm, IEEE Geosci. Remote Sens. Lett. 9 (2) (2012) 297–301.

[8] V.L.S. Silva, A.A. Emerick, P. Couto, J.L.D. Alves, History matching and production optimization under uncertainties–Application of closed-loop reservoir management, J. Pet. Sci. Eng. 157 (2017) 860–874.

[9] K. Zhang, L. Zhang, J. Yao, Y. Chen, R. Lu, Water flooding optimization with adjoint model under control constraints, J. Hydrodyn. 26 (1) (2014) 75–85.

[10] H. Rosenbrock, An automatic method for finding the greatest or least value of a function, Comput. J. 3 (3) (1960) 175–184.

[11] Y.-W. Shang, Y.-H. Qiu, A note on the extended Rosenbrock function, Evol. Comput. 14 (1) (2006) 119–126.

[12] S. Mahdavi, M.E. Shiri, S. Rahnamayan, Metaheuristics in large-scale global continues optimization: A survey, Inform. Sci. 295 (2015) 407–428.

[13] A. LaTorre, S. Muelas, J.-M. Peña, A comprehensive comparison of large scale global optimizers, Inform. Sci. 316 (2015) 517–549.

[14] H.Y. Benson, D.F. Shanno, R.J. Vanderbei, A comparative study of large-scale nonlinear optimization algorithms, in: High Performance Algorithms and Software for Nonlinear Optimization, Springer, 2003, pp. 95–127.

[15] W.W. Hager, D.W. Hearn, P.M. Pardalos, Large Scale Optimization: State of the Art, Springer Science & Business Media, 2013.

[16] S. Jaroslaw, R.T. Haftka, Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments, NASA Langley Technical Report Server, 1996, pp. 1–23.

[17] M.Z. Ali, N.H. Awad, P.N. Suganthan, Multi-population differential evolution with balanced ensemble of mutation strategies for large-scale global optimization, Appl. Soft Comput. 33 (2015) 304–327.

[18] T. Bhowmik, H. Liu, Z. Ye, S. Oraintara, Dimensionality reduction based optimization algorithm for sparse 3-D image reconstruction in diffuse optical tomography, Sci. Rep. 6 (2016) 22242.

[19] Z. Yang, K. Tang, X. Yao, Large scale evolutionary optimization using cooperative coevolution, Inform. Sci. 178 (15) (2008) 2985–2999.

[20] M.A. Potter, K.A. De Jong, A cooperative coevolutionary approach to function optimization, in: International Conference on Parallel Problem Solving from Nature, Springer, 1994, pp. 249–257.

[21] F. van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 225–239.

[22] J. Liu, K. Tang, Scaling up covariance matrix adaptation evolution strategy using cooperative coevolution, in: International Conference on Intelligent Data Engineering and Automated Learning, Springer, 2013, pp. 350–357.

[23] Y. Liu, X. Yao, Q. Zhao, T. Higuchi, Scaling up fast evolutionary programming with cooperative coevolution, in: Proceedings of the 2001 Congress on Evolutionary Computation, Vol. 2, IEEE, 2001, pp. 1101–1108.

[24] T. Jansen, R.P. Wiegand, The cooperative coevolutionary (1+ 1) EA, Evol. Comput. 12 (4) (2004) 405–434.

[25] D. Sofge, K. De Jong, A. Schultz, A blended population approach to cooperative coevolution for decomposition of complex problems, in: Proceedings of the 2002 Congress on Evolutionary Computation, Vol. 1, IEEE, 2002, pp. 413–418.

[26] R. Salomon, Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms, BioSystems 39 (3) (1996) 263–278.

[27] W. Chen, T. Weise, Z. Yang, K. Tang, Large-scale global optimization using cooperative coevolution with variable interaction learning, in: International Conference on Parallel Problem Solving from Nature, Springer, 2010, pp. 300–309.

[28] M.N. Omidvar, X. Li, Y. Mei, X. Yao, Cooperative co-evolution with differential grouping for large scale optimization, IEEE Trans. Evol. Comput. 18 (3) (2014) 378–393.

[29] K. Tang, X. Li, P.N. Suganthan, Z. Yang, T. Weise, Benchmark functions for the CEC'2010 special session and competition on large-scale global optimization, Tech. Rep., Nature Inspired Computation and Applications Laboratory, 2009.

[30] X. Li, K. Tang, M.N. Omidvar, Z. Yang, K. Qin, Benchmark functions for the CEC'2013 special session and competition on large-scale global optimization, Tech. Rep., RMIT University, Melbourne, Australia, 2013.

[31] M.N. Omidvar, X. Li, X. Yao, Cooperative co-evolution with delta grouping for large scale non-separable function optimization, in: IEEE Congress on Evolutionary Computation (CEC), IEEE, 2010, pp. 1–8.

[32] I.D. Falco, A.D. Cioppa, G.A. Trunfio, Investigating surrogate-assisted cooperative coevolution for large-scale global optimization, Inform. Sci. 482 (2019) 1–26.

[33] X.-M. Hu, F.-L. He, W.-N. Chen, J. Zhang, Cooperation coevolution with fast interdependency identification for large scale optimization, Inform. Sci. 381 (2017) 142–160.

[34] M.N. Omidvar, M. Yang, Y. Mei, X. Li, X. Yao, DG2: A faster and more accurate differential grouping for large-scale black-box optimization, IEEE Trans. Evol. Comput. 21 (6) (2017) 929–942.

[35] Y. Sun, M. Kirley, S. Halgamuge, A recursive decomposition method for large scale continuous optimization, IEEE Trans. Evol. Comput. 22 (5) (2018) 647–661.

[36] M.N. Omidvar, X. Li, K. Tang, Designing benchmark problems for large-scale continuous optimization, Inform. Sci. 316 (2015) 419–436.

[37] Y. Mei, M.N. Omidvar, X. Li, X. Yao, A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization, ACM Trans. Math. Softw. 42 (2) (2016) 13.

[38] P.L. Toint, Test problems for partially separable optimization and results for the routine PSPMIN, FNDP, Namur Report (83/4), 1983.

[39] Z. Yang, K. Tang, X. Yao, Multilevel cooperative coevolution for large scale optimization, in: IEEE Congress on Evolutionary Computation (CEC), IEEE, 2008, pp. 1663–1670.

[40] M.N. Omidvar, X. Li, Z. Yang, X. Yao, Cooperative co-evolution for large scale optimization through more frequent random grouping, in: IEEE Congress on Evolutionary Computation (CEC), 2010, pp. 1–8.

[41] T. Ray, X. Yao, A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning, in: IEEE Congress on Evolutionary Computation (CEC), IEEE, 2009, pp. 983–989.

[42] E. Sayed, D. Essam, R. Sarker, Dependency identification technique for large scale optimization problems, in: IEEE Congress on Evolutionary Computation (CEC), IEEE, 2012, pp. 1–8.

[43] Y.-j. Shi, H.-f. Teng, Z.-q. Li, Cooperative co-evolutionary differential evolution for function optimization, Adv. Nat. Comput. (2005) 428.

[44] Y. Sun, M. Kirley, S.K. Halgamuge, Extended differential grouping for large scale global optimization with direct and indirect variable interactions, in: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, ACM, 2015, pp. 313–320.

[45] Y. Sun, M.N. Omidvar, M. Kirley, X. Li, Adaptive threshold parameter estimation with recursive differential grouping for problem decomposition, in: Proceedings of the Genetic and Evolutionary Computation Conference, ACM, 2018, pp. 889–896.

[46] Y. Sun, X. Li, A. Ernst, M.N. Omidvar, Decomposition for large-scale optimization problems with overlapping components, in: IEEE Congress on Evolutionary Computation (CEC), 2019, pp. 326–333.

[47] M. Munetomo, D.E. Goldberg, Identifying Linkage by Nonlinearity Check, IlliGAL Report 98012, 1998.

[48] C. Liang, C. Chung, K. Wong, X. Duan, Parallel optimal reactive power flow based on cooperative co-evolutionary differential evolution and power system decomposition, IEEE Trans. Power Syst. 22 (1) (2007) 249–257.

[49] D. Komlen, D. Jakobović, Investigation of coevolutionary approach in gene regulatory network inference, in: 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2013, pp. 981–987.

[50] G.D. Chen, K. Zhang, X.M. Xue, L.M. Zhang, J. Yao, H. Sun, L. Fan, Y.F. Yang, Surrogate-assisted evolutionary algorithm with dimensionality reduction method for water flooding production optimization, J. Pet. Sci. Eng. 185 (1) (2020) 106633.

[51] I.E. Sutherland, R.F. Sproull, R.A. Schumacker, A characterization of ten hidden-surface algorithms, ACM Comput. Surv. 6 (1) (1974) 1–55.

[52] J. Liu, Y. Chen, J.M. Maisog, G. Luta, A new point containment test algorithm based on preprocessing and determining triangles, Comput. Aided Des. 42 (12) (2010) 1143–1150.

[53] N. Hansen, S. Kern, Evaluating the CMA Evolution Strategy on Multimodal Test Functions, Springer Berlin Heidelberg, 2004, pp. 282–291.

[54] M. Yang, M.N. Omidvar, C. Li, X. Li, Z. Cai, B. Kazimipour, X. Yao, Efficient resource allocation in cooperative co-evolution for large-scale global optimization, IEEE Trans. Evol. Comput. 21 (4) (2017) 493–505.

[55] N. Hansen, The CMA Evolution Strategy: A Comparing Review, Springer Berlin Heidelberg, 2006, pp. 75–102.

[56] A. LaTorre, S. Muelas, J.-M. Peña, Large scale global optimization: Experimental results with mos-based hybrid algorithms, in: IEEE Congress on Evolutionary Computation (CEC), IEEE, 2013, pp. 2742–2749.
[57] D. Molina, M. Lozano, F. Herrera, MA-SW-chains: Memetic algorithm based on local search chains for large scale continuous global optimization, in: IEEE Congress on Evolutionary Computation (CEC), IEEE, 2010, pp. 1–8.
[58] V.D.B. Frans, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 225–239.
[59] Y. Chen, D.S. Oliver, D. Zhang, Efficient ensemble-based closed-loop production optimization, SPE J. 14 (04) (2009) 634–645.
[60] L.M. Zhang, C.Y. Cui, X.P. Ma, Z.X. Sun, F. Liu, K. Zhang, A fractal discrete fracture network model for history matching of naturally fractured reservoirs, Fractals 27 (01) (2019) 1–15.
[61] G.D. Chen, K. Zhang, L.M. Zhang, X.M. Xue, D.Z. Ji, C.J. Yao, J. Yao, Y.F. Yang, Global and local surrogate-model-assisted differential evolution for waterflooding production optimization, SPE J. 25 (01) (2020) 105–118.
[62] Z. Yang, K. Tang, X. Yao, Self-adaptive differential evolution with neighborhood search, in: IEEE Congress on Evolutionary Computation (CEC), IEEE, 2008, pp. 1110–1116.