Dynamic Cooperative Coevolution for Large Scale Optimization

Xin-Yuan Zhang, *Student Member, IEEE*, Yue-Jiao Gong, *Member, IEEE*, Ying Lin, *Member, IEEE*, Jie Zhang, *Member, IEEE*, Sam Kwong, *Fellow, IEEE*, and Jun Zhang, *Fellow, IEEE*

***Abstract*—The cooperative coevolution (CC) framework achieves a promising performance in solving large scale global optimization problems. The framework encounters difficulties on nonseparable problems, where variables interact with each other. Using the static grouping methods, variables will be theoretically grouped into one big subcomponent, whereas the random grouping strategy endures low efficiency. In this paper, a dynamic CC framework is proposed to tackle the challenge. The proposed framework works in a computationally efficient manner, in which the computational resources are allocated to a series of elitist subcomponents consisting of superior variables. First, a novel estimation method is proposed to evaluate the contribution of variables using the historical information of the best overall fitness. Based on the contribution and the interaction information, a dynamic grouping strategy is conducted to construct the dynamic subcomponent that evolves in the next evolutionary period. The constructed subcomponents are different from each other, and therefore the required parameters to control the optimization of each subcomponent vary a lot in each evolutionary period. A stage-by-stage parameter adaptation strategy is proposed to adapt the optimizer to the dynamic optimization environment. Experimental results indicate that the proposed framework achieves competitive results compared with the state-of-the-art CC frameworks.**

***Index Terms*—Cooperative coevolution (CC), dynamic grouping (DyG) strategy, large scale global optimization (LSGO), nonseparable problems.**

Manuscript received May 28, 2018; revised September 25, 2018 and November 18, 2018; accepted January 15, 2019. Date of publication January 28, 2019; date of current version November 27, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61873095, Grant 61772569, Grant 61873097, and Grant U1701267, and in part by the Science and Technology Planning Project of Guangdong Province under Grant 2014B050504005. *(Corresponding authors: Yue-Jiao Gong; Jun Zhang.)*

X.-Y. Zhang is with School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China.

Y.-J. Gong and J. Zhang are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, South China University of Technology, Guangzhou 510006, China (e-mail: gongyuejiao@gmail.com; junzhang@ieee.org).

Y. Lin is with the Department of Psychology, Sun Yat-sen University, Guangzhou 510006, China.

J. Zhang is with the School of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China.

S. Kwong is with the Department of Computer Science, City University of Hong Kong, Hong Kong.

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author. The supplementary file contains three parts to elaborate on the performance of the proposed DCC: 1) Sensitivity Study of the Parameters; 2) Ablation Study; 3) Summary of the Experimental Results. The total size of the file is 404 KB.

Digital Object Identifier 10.1109/TEVC.2019.2895860

I. Introduction

IN THE era of Big Data, the dimensional space of problems becomes extremely large due to the growing volume, velocity, and variety of the data [1]. Large scale global optimization (LSGO) problem is a crucial component in many Big Data applications. These problems involve hundreds or even thousands of variables, which arise in the areas ranging from industrial design [2], [3] to social service [4], [5].

Literature on evolutionary computation [6], [7] shows that it is a powerful and prevalent optimization technique. For small or medium scale optimization problems, evolutionary algorithms (EAs) achieve a huge success in numerous kinds of applications [8]–[10]. However, when the problem dimensionality goes beyond the search ability of traditional EAs (EAs are commonly developed for problems of dimension lower than 100 [11]–[13]), it becomes a challenge to achieve satisfactory solutions in terms of solution quality and convergence rate [14].

There commonly exist two ways to tackle the curse of dimensionality challenge, one is designing more powerful LSGO algorithms [15]–[17], while the other is to perform dimension reduction. The motivations of the two methods are totally different. The former method focuses on how to enhance the search ability in the high-dimensional space of the problem. It considers all variables as a whole and develops appropriate evolving strategies required in LSGO, e.g., the strategy to enhance the population diversity. Compared with the former approach, the later uses the divide-and-conquer mechanism to achieve the dimension reduction, which is a more flexible framework in terms of scalability and parallelizability [18], [19]. Owing to the remarkable abilities of the human cooperation in the social development, the cooperative manner is prevalent in the design of the optimization algorithms. In this paper, we focus on the improvement of optimization algorithm when using cooperation strategy. The cooperative coevolution (CC) framework transforms the original problem into a number of small scale subproblems, and solves them separately. Finally, the local feasible solutions are assembled into a global one. An early attempt to this line of thinking is the CC framework developed in [22]. In the traditional CC frameworks, the decision variables are pregrouped into different subcomponents by specific grouping methods. For separable LSGO problems, variables can be divided into a set of disjoint subcomponents [24]. Each subcomponent represents an independent subproblem. The optimization process of each subproblem does not influence each other. However,

real world LSGO problems can be nonseparable [23]. In this case, variables are theoretically nonseparable, and hence it is difficult to perform dimension reduction.

To the best of our knowledge, the literature of the methods that address the nonseparable LSGO problems is rather limited. The random grouping mechanism [22] has some effects, but it endures low efficiency due to its randomness. Therefore, in order to expand the usage of CC framework to nonseparable LSGO problems, we propose a dynamic CC (DCC) framework in this paper. The main contributions are given as follows.

1) *A DCC Framework for the Nonseparable LSGO Problems:* DCC decomposes LSGO problems according to the contribution of variables as well as their interactive relationship, which is a totally different mechanism compared with the existing static grouping methods. The nonseparable LSGO problems become separable in the proposed DCC framework, and hence the divide-and-conquer mechanism is applicable.
2) *A Dynamic Computational Resources Allocation (CRA) Strategy:* DCC dynamically constructs a series of subcomponents. Each subcomponent consists of two types of members, i.e., the basic members and the affiliated members. A prespecified number of variables are first determined to be the basic members if they have high contribution. After that, affiliated members are chosen from those who directly interact with the basic members. Computational resources are dynamically allocated to the constructed subcomponents, which makes DCC work in a computationally efficient manner.
3) *A History-Assisted Contribution Estimation Method:* For nonseparable LSGO problems, the contribution of each specific variable interacts with the other variables. Therefore, it is hard to compute the contribution of each single variable. In order to tackle this challenge, a novel history-based estimation method is proposed to evaluate the contribution of variables. In each evolutionary period of DCC, a few variables constitute a subcomponent and thus the improvement of the global objective value comes from the evolution of these variables. Considering this, DCC records the historical information of the objective improvements and their associated variables in each generation. The contribution of each variable is then estimated by its averaged contribution over a few generations in the history. The proposed estimation method makes the contribution information of variables countable and offers a promising thread of designing new grouping strategies.
4) *A Stage-by-Stage Parameter Adaptation (SSPA) Strategy:* The subcomponents of DCC framework are dynamically constructed, and they are different from each other. Therefore, the solution subspace in DCC is dynamically changed during the optimization. Under this circumstance, the sudden change of the solution subspace challenges the traditional parameter adaptation methods that perform the adaptation strategies gradually at low frequencies. To solve the problem, we further develop an SSPA strategy that is more suitable for our DCC framework. The strategy enhances the search efficiency of the optimizer.

Apart from the nonseparable LSGO problems, DCC achieves higher efficiency than the existing CC frameworks in solving fully separable LSGO problems. Due to the fully irrelevant structure, the superiority of variables is determined only by their contribution. Instead of optimizing separable variables one by one or as a whole, we dynamically construct the elitist subcomponents to undergo evolutionary optimization.

The performance of DCC is verified by four types of experiments and the experimental results indicate that it achieves promising results in terms of the solution quality and the convergence. The rest of this paper is organized as follows. Section II introduces the formulation of LSGO problems and the related work of the CC frameworks. The proposed DCC framework is described in Section III. In Section IV, the DCC framework is comprehensively investigated by comparison with state-of-the-art algorithms. Finally, the conclusions are drawn in Section V.

II. Related Techniques

In this section, first, we introduce and define different types of LSGO problems, including separable and nonseparable ones. After that, numerous decomposition methods are summarized.

A. LSGO Problems

Real world LSGO problems can be divided into two categories (separable and nonseparable) according to the interaction relationship among variables. Furthermore, there exist direct and indirect interactive relationships between any two variables. Based on the above mentioned attributes, there exist four types of LSGO problems: 1) fully separable LSGO problem; 2) partially separable LSGO problem; 3) overlapping LSGO problem; and 4) fully nonseparable LSGO problem. The separable LSGO problem is defined as

$$\arg\min_{\mathbf{x}} f(\mathbf{x}) = \left(\arg\min_{\mathbf{x}_1} f(\mathbf{x}_1, \ldots), \ldots, \arg\min_{\mathbf{x}_m} f(\ldots, \mathbf{x}_m)\right) \tag{1}$$

where $\mathbf{x} = (x^1, \ldots, x^D)$ represents the overall decision vector with D variables. The set $\mathbf{s} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ indicates the m disjoint subcomponents of $\mathbf{x}$. Given any subcomponents $\mathbf{x}_i$ and $\mathbf{x}_j$, we have $\mathbf{x}_i \cap \mathbf{x}_j = \phi$. For a separable LSGO function $f(\mathbf{x}), f(\mathbf{x})$ is fully separable if $m = D$, and $f(\mathbf{x})$ is partially separable if $1 < m < D$. If $m = 1$ and the variables of $\mathbf{x}$ directly interact with each other, $f(\mathbf{x})$ is called fully nonseparable LSGO function. Overlapping LSGO problems are a special type of nonseparable LSGO problem, which is defined as

$$f(\mathbf{x}) = F(f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_m)) \tag{2}$$

where function $F(f(\mathbf{x}_i), f(\mathbf{x}_j))$ represents the combination of $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$, e.g., linear combination. Given two subcomponents $\mathbf{x}_i$ and $\mathbf{x}_j$, they are called overlapping subcomponents if $\mathbf{x}_i \cap \mathbf{x}_j = \mathbf{x}_o (\mathbf{x}_o \neq \emptyset)$. For any variable $\mathbf{x}^k \in \mathbf{x}_o$, $\mathbf{x}^k$ interacts with variables of $\mathbf{x}_i$ and $\mathbf{x}_j$ at the same time.

In order to investigate the generalization of the LSGO algorithms, benchmark functions are desired to simulate different types of real world LSGO problems. Therefore, benchmark functions, such as CEC2010 [27] and CEC2013 [28], are developed to provide a suitable evaluation platform of the LSGO algorithms. Each type of the LSGO benchmark function represents a specific type of the real world LSGO problem. In the CEC'2013 benchmark functions, the LSGO function $f(\mathbf{x})$ is defined as

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}_i) \tag{3}$$

where subcomponents $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ are combined using the weighted linear combination.

B. Traditional Cooperative Coevolution Frameworks

Owing to the evolutionary struggles, the cooperation strategy becomes a unique tool of the human beings [30]. The cooperation theory is promoting the development of human beings, such as the successful vaccination [31], [32]. The coevolutionary games are reviewed in [33]. An early attempt of the design of generic divided-and-conquer framework is conducted by [34]. A CC genetic algorithm is proposed to divide the original D-dimensional problem into D subcomponents, and each subcomponent is solved by a subpopulation in a round-robin fashion. In the traditional CC frameworks, the performance of this divide-and-conquer strategy depends on whether a decomposition method is appropriate for a specific type of the LSGO problem. A number of decomposition strategies are designed to divide the LSGO problem into a set of subcomponents. The existing decomposition methods of CC framework can be classified into two categories: 1) static grouping strategy and 2) dynamic grouping (DyG) strategy.

As for the static grouping strategy, the decision variables are divided into numerous subcomponents by a deterministic decomposition procedure and the subcomponents remain unchanged over the whole optimization process. In the cooperative particle swarm optimization (CPSO) [35], the decision variables are randomly split into k parts ($k << D$) and the k parts remain unchanged in the following optimization process. Using the random decomposition strategy is not quite accurate, because there are interactions between decision variables. The CC with variable interaction learning (CCVIL) is an early attempt to group interacting variables into subcomponents by the perturbation technique [36] on LSGO problems. CCVIL consists of two stages, i.e., interaction detection and optimization of a set of reduced space, which becomes a classic procedure of traditional CC. In order to improve the grouping efficiency, differential grouping (DG) strategy is proposed in [24], which is derived from the problem formulation in Subsection A. DG divides variables interacting with each other into a same subcomponent by a deterministic interaction detection operator. The DG strategy achieves high grouping accuracy on CEC'2010 large scale benchmark functions. However, the original DG strategy encounters difficulties in terms of grouping accuracy and effectiveness when new attributes of high-dimensional functions are introduced. Specifically, for the overlapping LSGO functions, DG fails to detect the indirect interactive relationship between decision variables, i.e., overlapping components. The extended DG [37] is an early attempt to address this issue. It merges two subcomponents with same decision variables and repeats this process until all subcomponents are totally independent with each other. Global DG (GDG) [38] deals with this issue by conducting breadth-first or depth-first search within the interaction structure matrix. In addition to the overlapping component issue, computational error caused by computer operations is also considered in GDG. Further, the DG2 [39] is an updated version of the original DG, which enhances grouping accuracy and efficiency by the parameters adaptation technique and the reusing of sampling points, respectively.

As can be noted from the above static grouping methods, they focus on how to transform the original problem space into a series of independent reduced subspaces using different decomposition (i.e., variable grouping) techniques. Concretely, the mechanism of the static grouping strategy works well when the high-dimensional problem is fully or partially separable. However, this is not the case for nonseparable problems. For nonseparable problems, decision variables directly or indirectly interacts with each other. Therefore, the original problem space is divided into a number of independent reduced subspaces. It is therefore desirable to design dynamic decomposition techniques to cope with this issue. In multilevel CC (MLCC) [42], the high-dimensional problem is decomposed by a series of specific designed decomposers and decomposers are selected according to their performance during the evolutionary process. A new CC framework is proposed in [40], unlike CPSO, it conducts random decomposition in every beginning of the evolution cycle, and the theoretical analysis is given to ascertain the effectiveness of this random grouping strategy. Li and Yao [41] proposed a CC-based PSO, where an adaptive scheme is adopted to dynamically determine the size of subcomponent. These DyG strategies are random methods and they achieve better performance than static grouping on nonseparable large scale optimization problems.

C. Computational Resource Allocation

There are two critical research points of CC-based algorithm, the one is the above-mentioned decomposition strategy, and another is how to allocate the limited computational resource to different subcomponents. The former focuses on how to accurately and efficiently transform the high-dimensional problem space into a number of reduced subspaces and ensure its effectiveness as high as possible. The decomposition strategy can be viewed as a preprocessing procedure of the optimization process. After decomposition, subcomponents are optimized in a round-robin mode under traditional CC framework. Recent research has shown the imbalance feature among subcomponents in terms of their contributions to the original LSGO problem [43]. Due to the imbalance among subcomponents, it will be an efficient way of balancing computational resources among different subcomponents. Along this line of thinking, various

contribution-based CC (CBCC) frameworks with resource allocation are proposed.

In [43], two CBCC (CBCC1 and CBCC2) algorithms are proposed, where the contributions of different subcomponents are quantified and the one with the biggest contribution is chosen to undergo the next evolutionary period. The experimental results show that the mechanism of CBCC obtains improved performance for high-dimensional problems with imbalanced subcomponents. However, on some problem instances, the existing CBCC variants still endure over exploration or over exploitation issue, which limit the solution quality. CBCC3 [44] inherits main part of CBCC, and balances the exploration and exploitation of CC optimizer, which achieves significant improvement over CBCC. Yang *et al.* [45] proposed a CC framework named CCFR and conduct resource allocation in a more efficient way according to the dynamic contributions of each subcomponent. Experimental results suggest that CCFR is capable of scheduling computational resources in a more efficient way than CBCC methods. The performance of above CBCC frameworks depends on the accuracy of grouping methods, because "contribution" is measured at the grain of subcomponent. For separable LSGO problems, the contribution of each subcomponent can be accurately computed if the grouping results are correct. However, it is impossible to divide a nonseparable LSGO problem into a series of disjoint subcomponents. It becomes a challenge of how to conduct a decomposition strategy and computational resource allocation. In order to alleviate this issue, an overlapped CC is proposed in [26], where variables with a strong impact on objective function are assigned more computational resource. The impact of each variable is tested by a deterministic delta-disturbance strategy, but the impact of each variable changes with the current local search area and it consumes extra fitness evaluations (FEs). The above-mentioned CBCC algorithms perform successfully in improving the performance of the traditional CC framework. On account of accurate grouping results, crucial subcomponents possess more computational resource and therefore achieve higher solution quality.

The existing divide-and-conquer frameworks show promising performance on separable LSGO problems in the literature owing to the accurate grouping strategy and contribution-based mechanism. However, they endure limitations on some other problems, such as nonseparable LSGO problems. As the important components of real world LSGO problems, there are no specially designed LSGO algorithms to deal with them. This paper proposes a generic DCC framework to cope with this type of LSGO problem. The mechanism of DCC will be described in the following section.

III. Proposed Method

A. Dynamic Co-Operative Coevolution Framework

This section shows the overall framework of the proposed DCC framework including three novel operations: 1) the estimation of the dimensional contribution (EDC) strategy; 2) the DyG method; and 3) the SSPA strategy. The idea behind DCC is that the original LSGO nonseparable problems are considered as a series of dynamic subcomponents for optimization.

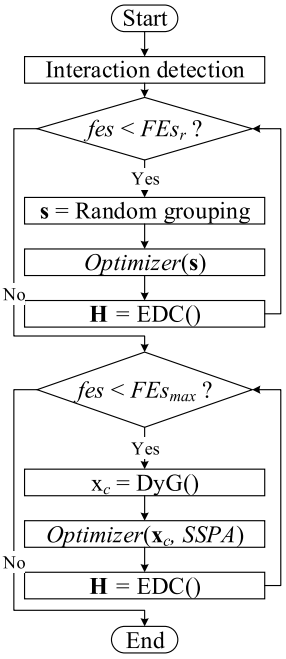

Fig. 1. Overall flowchart of DCC. The DyG, EDC and SSPA represent the dynamic grouping operation, the estimation of dimension contribution, and the stage-by-stage parameters adaptation, respectively. The algorithm terminates at a given number of the fitness evaluations.

Each subcomponent is dynamically constructed during the optimization process consisting of numerous superior variables. The superiority of a variable depends on its contribution and the interaction relationship with other variables. The EDC strategy is first conducted to evaluate the contribution of each variable according to the historical information of the best overall fitness value, which will be introduced in detail in the following Subsection B. After the EDC operation, the dynamic subcomponent is constructed by the DyG method. In DyG, the superiority of variables is ranked according to their contribution information and interaction information. In static grouping methods, subcomponents are predetermined and unchanged during the whole optimization process. The optimization of the same subcomponent is continuous. However, this is not the case for DyG methods. The new constructed dynamic subcomponent can be different from the previous dynamic subcomponents. Therefore, parameters of the previous optimizers become insignificant for the following optimization process. An SSPA strategy is proposed in order to ensure the effectiveness of the parameter adaptation strategy.

Fig. 1 illustrates the overall flowchart of DCC, which consists of two steps. First, a random grouping-based CC algorithm is conducted as a startup phase in order to collect the variable contribution information. Then, the most superior subcomponent is determined by DyG, and this subcomponent is optimized by the optimizer with SSPA strategy. After the optimization of this subcomponent, the contribution information of this subcomponent is collected using the EDC strategy. Based on the new contribution information, the most promising subcomponent will be generated again by DyG. Pseudocode of this DCC framework is given in

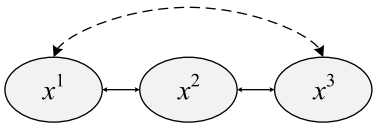

Fig. 2. Variable interaction relationship. The dashed and solid arrows represent the indirect and direct interaction between any two variables.

Algorithm 1 DCC Framework(f, D, $\boldsymbol{P}$, *Optimizer*())

```
Procedure DCC(f, D, P, Optimizer())
  Interaction matrix Λ = DG2 (f, D);
  while (fes < FES_r) do
    {x_1, x_2, ..., x_M} ← Random Grouping (f, D);
    for k = 1 → M do
      (P) ← optimizer(f, D, FES = |P|, non-PA)
      update H;                                   // EDC
      k ← k + 1;
    end for
    fes ← fes + |P|;
  end while
  while fes < FES_m do
    for g = 1 → G_d do
      Computational resource allocation(f, D, H);        // DyG
      (P) ← optimizer(f, D, FES = |P| · G_o, SSPA);  // SSPA
      update H;
      fes ← fes + |P|· G_d;
    end for
  end while
end procedure
```

Algorithm 1. The details of EDC, DyG, and SSPA strategies are given in the following subsections.

B. Estimation of Dimensional Contribution

The approximation accuracy of the contribution information influences the efficiency of CRA. In this section, we first analyze the difficulty of using the existing contribution-based techniques on the nonseparable LSGO problems, and then we introduce the execution details of the EDC method.

The challenge of using a contribution-based mechanism in solving nonseparable LSGO problems is derived from the fully interactive structure of nonseparable LSGO problems. The interaction relationship between any two decision variables is a critical factor that determines the interaction structure of an LSGO problem, and it is hence a crucial criterion of conducting decomposition strategy. There are two types of variable interaction relationships which are depicted in Fig. 2. If variable x^1 and variable x^3 directly interact with variable x^2, the relationship between x^1 and x^3 is called indirect interaction [37]. Let the LSGO problem $f(\mathbf{x}), \mathbf{x} = (x^1, x^2, \ldots, x^9)$, be a typical partially separable LSGO problem, the interactive graph of decision variables is depicted in Fig. 3(a). Variable x^i and x^j will be connected by the two-headed solid arrows if they directly interact with each other. In this case, the interaction graph is an unconnected graph, and it can be divided in to three connected subcomponents, i.e., $\mathbf{x}_1 = \{x^1, x^2, x^3\}$, $\mathbf{x}_2 = \{x^4, x^5, x^6\}$, and $\mathbf{x}_3 = \{x^7, x^8, x^9\}$. The subcomponents $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ are independent and the contribution of each subcomponent can be evaluated independently. The existing CBCC frameworks evaluate the superiority of each subcomponent according to their contribution and interactive relationship. After that, computational resources are allocated to the subcomponents with highest superiority (SHS). Therefore, the existing CBCC frameworks achieve success in solving partially separable LSGO problems owing to the special interactive structure. For nonseparable LSGO problems, decision variables interact with each other. In Fig. 3(b), the interaction graph is a connected graph in which the two-headed dashed arrows represent the indirectly interactive relationship. In this case, there is no deterministic partition strategy to divide the connected graph into independent subcomponents. For static grouping methods, DG divides the connected graph using the greedy strategy, whilst DG2 considers the connected graph as a single component. As for random grouping methods, the connected graph is randomly partitioned. The subcomponents obtained by DG and random grouping methods are independent. Therefore, it is difficult to evaluate the contribution of the subcomponents. In order to tackle this issue, we propose a novel method to

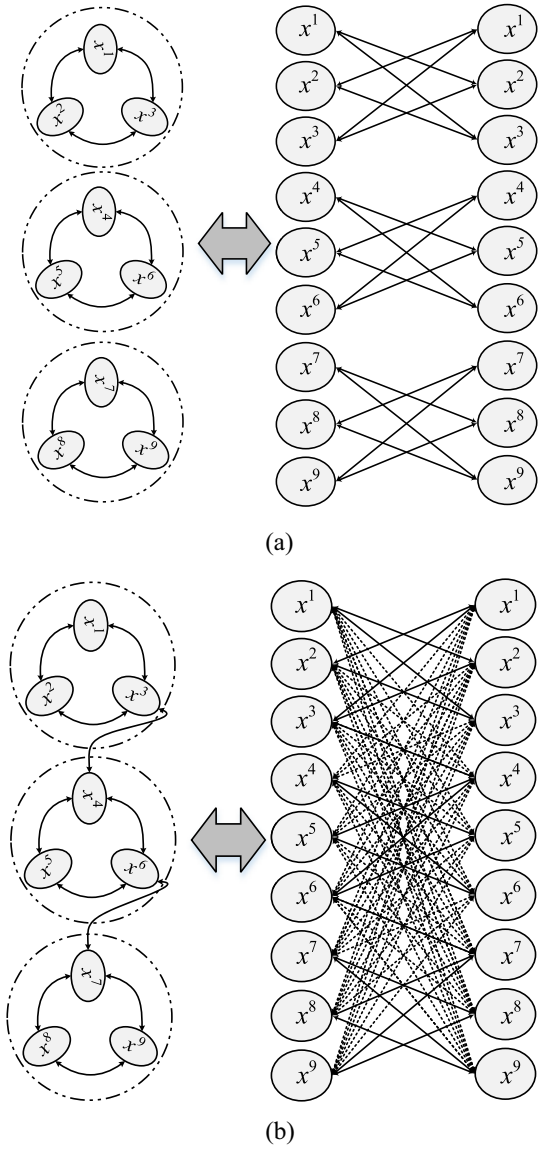

Fig. 3. Interaction graph of (a) partially separable LSGO function and (b) overlapping LSGO function. The dashed and solid arrows represent the indirect and direct interaction between any two variables.

estimate the contribution of variables. If the contribution criterion is capable of ranking the impact of subcomponents, it is not necessary to obtain the accurate contribution of subcomponents. Along this line of thinking, we propose an EDC method. EDC estimates the contribution of each single variable using the historical information of the best overall fitness.

Let $\mathbf{H} = (\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_D)$ be the contribution vector archiving historical contribution of each variable. Vector $\mathbf{H}_i = (H_{i,1}, H_{i,2}, \ldots, H_{i,t})$ tracks the contribution of the ith variable x^i, where index t represents that the ith variable is selected into SHS for the tth time. To make it concrete, we assume that the current subcomponent is $\mathbf{x}_c = \{x^{c,1}, x^{c,2}, \ldots, x^{c,T}\}$. After the optimization of $\mathbf{x}_c$, we calculate the improvement of the best overall objective value ΔF as (4), where f_{lbest} and f_{best} represent the last and the current best overall objective value, respectively. The current contribution of x^i is computed as (5). Factor λ_i represents that whether the ith variable is a member of $\mathbf{x}_c$. If the ith variable x^i belongs to $\mathbf{x}_c$, λ_i is set to 1, otherwise, λ_i is set to 0, as shown in (6). Owing to the parameter setting of λ_i, the contribution of the variables within the current SHS are updated

$$\Delta F = |f_{\text{best}} - f_{\text{lbest}}| \tag{4}$$

$$H_{i,t+1} = \lambda_i \cdot \Delta F \tag{5}$$

$$\lambda_i = \begin{cases} 1 & \text{if } x^i \in \mathbf{x}_c \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

C. Dynamic Grouping

Divide-and-conquer mechanism is a crucial and effective way of solving the large scale optimization problems. The existing grouping methods achieves significant success when adopted to solve the fully separable or the partially separable LSGO problems. However, they encounter difficulties on the nonseparable LSGO problems. As for static grouping strategies, all variables of nonseparable LSGO problems are grouped into a big single group even if the majority of variables are not directly interacted with each other. Therefore, the divide-and-conquer mechanism is not applicable for nonseparable LSGO problems. For random grouping strategy, variables are randomly divided into numerous subcomponents with the same probability. The interaction information is not adopted in random grouping strategy, which is not an efficient way of conducting dimension reduction.

Based on the above-mentioned challenges, we propose a DyG strategy considering both the interaction and contribution information of variables. At the beginning of DCC, variables are randomly divided into numerous subcomponents and each subcomponent is optimized with specific predetermined generations. In the startup stage, the contribution vector $\mathbf{H}$ is initialized. After the startup stage, there is sufficient contribution information so as to support DyG. We first transform the contribution vector $\mathbf{H}$ into an averaged contribution vector $\mathbf{A} = (A_1, A_2, \ldots, A_D)$, where A_i represents the mean contribution of the ith variable over the $|A_i|$ evolutionary periods. The calculation of the vector $\mathbf{A}$ is given as (7), where $H_{i,j}$ represents the jth evolutionary period of the ith variable

$$A_i = \frac{1}{|H_i|} \sum_{j=1}^{|H_i|} H_{i,j}. \tag{7}$$

Algorithm 2 Dynamic Grouping (f, D, $\mathbf{H}$)

```
1:  Procedure Dynamic Grouping (f, D, H)
2:      averaged contribution vector A ← 0;
3:      dynamic group x_c ← ∅;
4:      for i = 1 → ||H|| do
5:          for j = 1 → ||H_i|| do
6:              A_i ← A_i + h_{i,j};
7:              j ← j + 1
8:          end for
9:      end for
10:         sort (A, descending);
11:     for i = 1 → N_1 do
12:         for j = 1 → N_2 do
13:             x^i ← variable index corresponding to A_i
14:             if x^i ∉ x_c then
15:                 x_c ← x_c ∪ x^i;
16:             end if
17:             for k = 1 → D do
18:                 x^k ← variable index interacting with x^i
19:                 if x^k ∉ x_c then
20:                     x_c ← x_c ∪ x^k;
21:                     break
22:                 end if
23:                 k ← k + 1
24:             end for
25:         end for
26:     end for
27: end procedure
```

Two integers (N_1 and N_2) are adopted to control the size of the SHS. SHS consists of basic members and affiliated members. We rank the variables in a descending order according to averaged contribution vector $\mathbf{A}$. The top N_1 variables $\{x^{r,1}, x^{r,2}, \ldots, x^{r,N1}\}$ are chosen as the basic members of the current SHS, i.e., $\mathbf{x}_c = \{x^{r,1}, x^{r,2}, \ldots, x^{r,N1}\}$. For each $x^{r,i}$ ($i \in \{1, 2, \ldots, N_1\}$), we denote its interactive variables by a linked list $L(x^{r,i})$. The computation of the interactive matrix uses the techniques proposed by [39]. The variables of $L(x^{r,i})$ are ranked in a descending order according to averaged contribution vector $\mathbf{A}$. We adopt the notation $x^{r,i,k}$ to represent the kth item of $L(x^{r,i})$. After that, we use a round-robin manner to select the affiliated members. Starting with the first basic member $x^{r,1}$, we add the affiliated variable $x^{r,1,1}$ into $\mathbf{x}_c$ and then eliminate $x^{r,1,1}$ from all linked list $L(x^{r,i})$ ($i \in \{1, 2, \ldots, N_1\}$). For the remaining basic members $x^{r,i}$ ($i \in \{2, 3, \ldots, N_1\}$), we find their affiliated variables and add them into $\mathbf{x}_c$. The above mentioned operation is repeated for N_2 times. The constructed $\mathbf{x}_c$ is adopted as the current SHS. The details of DyG are given in Algorithm 2. The construction of SHS is hot on the heels of the sorting operation, which is a two nested loop (step 11 to step 26). The outer loop selects variables with highest averaged contribution whilst inner loop selects variables interacting with the variables selected by the outer loop. The constructed SHS is the subcomponent that is ready for the next optimization. As can be noted from DyG, the computational resource is allocated to variables with high contribution and their interactive variables. DyG considers both contribution and interaction information and therefore

Algorithm 3 Optimizer(*f*, *D*, *FEs*, *SSPA*)

```
while fes < FEs do
    if fes equals 1 then
        initialize all parameters;
    end if
    Initialize P using x_lbest;
    for i = 1 → NP do
        update P_i;
        Evaluate P_i;
        fes ← fes +1
        i ← i + 1;
    end for
    PA;
end while
```

becomes more efficient when solving large scale nonseparable problems (especially on overlapping LSGO problems).

D. Stage by Stage Parameters Adaptation Principle

Parameter adaptation (PA) is an effective way of improving the performance of optimizers. For CC-based LSGO frameworks, SaNSDE [25] is widely used as the optimizer and achieves promising performance. In SaNSDE, the scale factor and crossover rate are self-adapted according to the improvement of the best overall value of objective function, which serves as a basis criterion in numerous parameter adaptation strategies. For separable LSGO problems, subcomponents are disjoint and the optimization process of the same subcomponent is continuous. However, for nonseparable LSGO problems, the subcomponents constructed by DyG are different from each other and interactive with each other. Improvement on the best overall value of objective function is meaningless for parameter adaptation when optimization is switched from one subcomponent to another subcomponent, since the two subcomponents represent two different subproblems.

Apart from the continuity issue, the PA of LSGO algorithms encounters another challenge. In order to give a fair chance for different subcomponents, computational resources allocated to optimizing each subproblem is not so much. Taking the contribution-based large scale optimization algorithms as an example. The prespecified number of evolutionary generations for each subcomponent is 100. However, the PA, such as the mutation related factor and the crossover related factor, are conducted for every 50 and 25 generations. If PA is independently conducted between optimizing different subcomponents, the frequency of PA will be pretty low. Therefore, it takes no significant effects in improving the evolutionary searching efficiency.

Due to the above reasons, in the proposed DCC, it is not suitable to employ the PA that is commonly used in the current LSGO algorithms. To the best of our knowledge, the description of PA of the LSGO algorithms is unclear. This paper proposes an SSPA strategy. A stage refers to the optimization of a dynamic subcomponent. In SSPA strategy: 1) PA is independently conducted within different stages and 2) PA is supposed to obtain high frequency. In order to satisfy the first principle, we initialize parameters at the beginning of each stage. PA is conducted according to the fitness improvement caused by the optimization of the current subcomponent. As for the second principle, it helps us to find the appropriate optimizer, e.g., JaDE [46]. In JaDE, PA is conducted after each evolutionary generation. If the dynamic subcomponent is assigned 100 evolutionary generations, PA will be conducted 100 times in this stage, which will be significantly higher than that of SaNSDE. The details of SSPA strategy is shown in Algorithm 3. SSPA ensures that PA is guided only by the searching behavior within the current subspace, and PA is conducted in a high frequency.

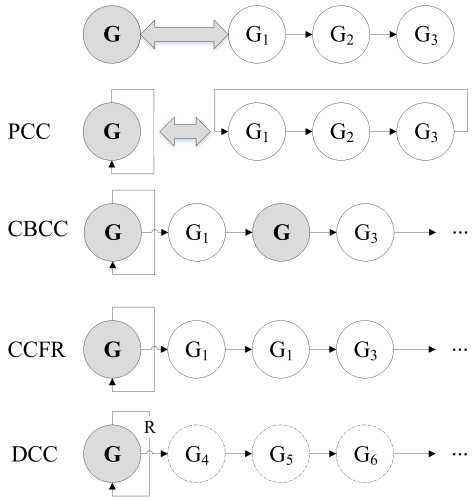

Fig. 4. Computational resource allocation of various CC framework including traditional CC, CBCC, CCFR, and DCC. The subgroup G_i ($i > 3$) is denoted by the dashed cycle, which represents the dynamic subgroup of DCC. For PCC, CBCC, and CCFR, the subgroups remain unchanged during the whole optimization process.

E. Comparison Between DCC and the Other CC Frameworks

In this section, we compare the difference of computation resource allocation between DCC and other CC frameworks. Fig. 4 shows the computational resource allocation methods of traditional pure CC (PCC), CBCC, CCFR, and DCC framework. To make it concrete, we assume that the overall LSGO problem is divided into three sub-subcomponents using a static grouping method, e.g., DG2. The solid gray cycle represents that three sub-subcomponents are optimized from subcomponent G_1 to subcomponent G_3. For the traditional CC framework, sub-subcomponents are optimized in a round-robin manner and each optimization process is limited to prespecified evolutionary generations. The computational resource is uniformly allocated to each sub-subcomponent in the traditional CC framework. CBCC adopts a test phase to collect the contribution information of each subcomponent. Assuming that subcomponent G_1 obtains the highest contribution after the test phase, G_1 wins the control of computational resource for evolutionary process. After the optimization of G_1 with prespecified generations, CBCC enters the test phase again. Compared with CBCC, CCFR works in a more efficient way, which conducts test phase only at the beginning of the whole optimization process. In CCFR, if the subcomponent G_1 achieves the biggest contribution, G_1 win the next optimization chance. After the optimization of G_1, the contribution of all sub components (G_1, G_2, and G_3) are changed

and the one with the biggest contribution (G_1) undergoes the next evolutionary process.

As can be noted from the above mentioned operation flowchart, the CBCC is able to dynamically select the most promising subcomponent and allocate more computational resources to them. The dynamic computational resource allocation mechanism makes the CCFR work efficiently in solving separable or partially separable LSGO problems. In Fig. 4, PCC, CBCC, and CCFR frameworks adopt static grouping strategy in which all subcomponents are predetermined before the optimization process. Different from above mentioned CC frameworks, DCC works in a more efficient way in dealing with LSGO nonseparable problems. Particularly, DCC allocates the computational resource to a series of dynamic subcomponents, which are depicted using the cycles with dashed line. Similar to the CBCC frameworks, DCC adopts a startup phase to collect the contribution of each variable. In the startup phase, random grouping strategy is conducted and the subcomponents are optimized in a round-robin manner. After the startup phase, a new subcomponent G_4 is constructed using the DyG operation, and G_4 achieves the next chance of optimization. In a similar way, G_5 and G_6 are new dynamic subcomponents for the next two optimization periods.

The divide-and-conquer mechanism plays a crucial role in solving the LSGO problems. The majority of the state-of-the-art large scale optimization algorithms are based on this mechanism and achieve success. There are two critical parts of implementing the divide-and-conquer mechanism: 1) the division strategy (for the "divide" process) and 2) the cooperative and coevolution (for the "conquer" process). This proposed algorithm improves both of the two parts, and the main contributions can be given as follows.

1) For the divide process, a dynamic division strategy for the nonseparable LSGO problems is proposed. For the cooperative and coevolution framework in solving LSGO problems, it is a critical issue to divide the large scale problems into a number of small scale problems. There are numerous division strategies to deal with the separable LSGO problems. The existing division strategies are based on the DG method. As for the nonseparable LSGO problems, variables are interactive with each other according the DG-based methods. There are no specific division strategies to decompose the nonseparable LSGO problems in addition to the random division strategies. In order to solve the LSGOs in a more efficient and effective way, we design a DyG method by which the LSGO problem is dynamically divided into small scale optimization problems. In the DyG method, the variables are divided according to their contribution to the LSGO problem. In this paper, the contribution of dimension is adopted to replace the gradient of each variable, which is supposed to point in the direction of the greatest rate of the decrease of the optimization problem. The computation of gradient consumes too many FEs, whilst the calculation of the contribution of dimension requires no additional evaluations.
2) For the conquer process, a novel CRA method is proposed. The existing CRA methods depend on the static division methods, i.e., DG [24] and DG2 [39]. The subgroups of variables are fixed during the whole optimization process. The computational resources are assigned to the static subgroups with biggest contribution in terms of the improving of the fitness function. In the proposed DCC, the computational resources are allocated to the dynamic subgroups with superior variables.

We propose DCC to branch the contribution-based concept out into nonseparable LSGO problems. Different from the existing static grouping methods or random grouping methods, we transform the nonseparable LSGO into a series of dynamic subcomponents. Along this line of thinking, we propose the EDC operation and DyG method to estimate the superiority of variables and construct the promising subcomponents, respectively. Due to the DyG mechanism, the constructed subcomponents are different from each other. An SSPA strategy is proposed to adapt the existing PA strategy into our framework.

IV. Experimental Results

A. Experimental Setup

In this section, experiments are conducted to ascertain the performance of DCC. The LSGO benchmark functions of CEC2010 and CEC2013 are widely used to test the performance of LSGO algorithms. CEC2010 contains 18 separable LSGO problems and 2 nonseparable problems, whilst CEC2013 contains 11 separable LSGO problems and 4 nonseparable problems. As can be noted from the benchmark combinations, the majority are separable LSGO problems, which may favor a portion of LSGO algorithms. Particularly, the recently proposed CC frameworks focus on static grouping strategy, and they achieve high solution quality on separable LSGO problems. Their performance is well investigated on the CEC2010 and CEC2013. However, we also notice that their performance deteriorates a lot on nonseparable LSGO problems. To our best knowledge, there are no LSGO CC frameworks addressing this issue. The proposed DCC framework focuses on nonseparable LSGO problems. However, because of the low number of nonseparable LSGO problems in the CEC2010 and CEC2013 test suits, it is difficult to test the proposed method sufficiently. Fortunately, CEC2013 provides a flexible framework to develop new nonseparable LSGO problems. We develop ten nonseparable functions following the framework of CEC2013. In CEC2013, function f_{12} to function f_{14} are overlapping functions. Functions f_{13} and f_{14} provide two different templates of overlapping LSGO problems. In order to obtain enough benchmark functions, ten overlapping functions are derived from f_{13} and f_{14}, represented by DF1–DF10. For simplicity, we adopt the way of designing overlapping functions with conforming subcomponents and conflicting subcomponents as f_{13}, and f_{14}, respectively. To be more concrete, DF1, DF2, DF3, DF4, and DF5 are derived from f_{13} by replacing the original base functions by Sphere, Elliptic, Rastrigin, Ackley, and Rosenbrock functions. Similarly, DF6, DF7, DF8, DF9, and DF10 are derived from F13 by replacing the original base functions by Sphere, Elliptic, Rastrigin, Ackley, and Rosenbrock functions.

TABLE I
PARAMETER SETTINGS OF DCC

Parameter	Description	Value
G_r	Number of evolutionary generations for optimization using random grouping strategy	800
G_d	Number of evolutionary generations for each optimization using dynamic grouping strategy	200
FEs_{max}	Maximum number of fitness evaluations	3×10^6
NP	Population size	50
N_1	Outer cycle number of DyG	100
N_2	Inner cycle number of DyG	$0.02D$
p	Control parameter of JaDE for mutation operation	0.1
c	Control parameter of JaDE for parameter adaptation	0.05

The base functions are the same as those of CEC2013. Default dimension of the above mentioned overlapping functions is set to 905. Apart from the nonseparable LSGO problems, the proposed DCC framework achieves high efficiency in terms of the CRA on the fully separable LSGO problems. In order to investigate the performance of DCC on the fully separable LSGO problems, the original LSGO functions f_1–f_3, f_{19}, and f_{20} of CEC2010 are adopted and denoted by OB01–OB03, OB19, and OB20. The original LSGO functions f_1-f_3, f_{12}, f_{13}, f_{14}, and f_{15} of CEC2013 are adopted and denoted by OA01–OA03, OA12, OA13, OA14, and OA15.

In Section IV-B, we compare DCC with the non-CBCC frameworks, including DECC-G, MLCC [42], DECC-DG [24], and FT-DNPSO [14], in order to investigate the effectiveness of DyG strategy. In Section IV-C, we compare DCC with the CBCC frameworks, such as CBCC [43], and CCFR, in order to ascertain the significance of the dynamic computational resource allocation. Apart from the overlapping functions, DCC achieves promising results on fully separable functions, which is studied in Section IV-D. Finally, experiments in Section IV-E are conducted to show the influence of different PA methods.

DCC and other large scale optimization algorithms are tested on the same platform with Intel core i7-7700 CPU running at 3.60 GHz, 8 GB memory, and Ubuntu 16.04.2 LTS 64-bit operating system. DCC adopts JADE [46] as optimizer since its PA mechanism satisfying the requirements of SSPA, i.e., high frequency of PA. The optimizer of DECC-G, MLCC, DECC-DG, CBCC, and CCFR is SaNSDE. Parameter settings are given in Table I. G_r and G_d are evolutionary generations used by random grouping and DyG, which are set to 800 and 200. The outer and inner cycle number of DyG are set to 100 and $0.02D$, respectively. For the compared algorithms, the group size is set to 50, which is suggested in [49]. The prespecified number of evolutionary generations is set to 200. The maximum FEs is set to 3×10^6 as suggested in [27] and [28]. Parameters settings of SaNSDE follow the default settings of the corresponding original paper. For CC-based algorithms (DECC-G, MLCC, DECC-DG, CBCC, CCFR), each subproblem is optimized for 100 evolutionary generations, which is suggested in [45]. The parameter settings of FT-DNPSO are recommended by [14]. The other parameter settings remain the same with the corresponding original papers.

All compared algorithms are conducted in 30 independent trials and the results are averaged over the trials. In all the four tables, two-sided Wilcoxon rank sum test is conducted at significance level $\alpha = 0.05$. In every bottom of the four tables, *w/t/l* represents that DCC achieves significantly superior, similar, and significantly inferior results than the compared algorithm.

B. Comparison With Noncontribution-Based CC Framework

In this section, comparison experiments are conducted in order to verify the performance of DCC in comparison with non-CBCC frameworks. The random grouping-based algorithms (DECC-G and MLCC), the DyG-based algorithm (FT-DNPSO), and the static grouping-based algorithm (DECC) are adopted in this paper. For DECC, we use DG other than DG2. Although DG2 achieves more accurate grouping results in CEC2013 benchmark functions than DG, DG2 is not appropriate in decomposition of overlapping functions. In DG2, all decision variables are interactive with each other and therefore divided into one single group. Thus, CC framework is not applicable in that case. In DG, indirectly interactive relationship between variables is not taken into consideration, and thus variables are divisible. Based on the above mentioned reasons, DG is adopted as a basic static grouping method in this paper.

Experimental results of DECC-G, DECC-DG, MLCC, and DCC are summarized in Table I. As can be noted from Table II, DCC significantly outperforms the other algorithms on almost all benchmark functions with several order of magnitude, functions DF01-DF02, DF05, and DF07-OB15.

In DECC-G and MLCC, random grouping method and adaptive random grouping methods are adopted. Random grouping mechanism is applicable in solving overlapping functions but pure random grouping method is in lack of efficiency. The reasons are given as follows. For a connected graph representing variable interaction information, there is no deterministic way of dividing the connected graph into a number of subcomponents. Although random grouping strategies is capable of increasing the probability of selecting two critical variables (e.g., interactive variables) into a same subcomponent, it is still not a straight way of using interaction information between variables. From the perspective of the CC evolutionary process, interaction relationship is not the only criterion that influences the search efficiency in solving overlapping function. The contribution of each variable is also a critical issue that is supposed to be considered. DECC-G and MLCC take no consideration of this contribution information. In the conclusion, the random grouping methods of DECC-G and MLCC are lack of guideline in terms of interaction information and contribution. In comparison, the DyG mechanism uses both the contribution and the interaction information. Therefore, DCC performs better than DECC-G and MLCC.

As for DECC-DG, it incorporates the DG method, which divides the fully connected interaction graph into a number

TABLE II
EXPERIMENT ON 12 OVERLAPPING FUNCTIONS AND 3 FULLY NONSEPARABLE FUNCTIONS IN COMPARISON WITH NONCONTRIBUTION-BASED ALGORITHMS. ALL THE ALGORITHMS ARE CONDUCTED IN 30 INDEPENDENT TRIALS. THE AVERAGE AND STANDARD DEVIATION VALUES OF THE RESULTS ARE REPRESENTED BY MEAN, AND STD., RESPECTIVELY. THE HIGHLIGHTED ENTRIES ARE SIGNIFICANTLY BETTER. WILCOXON RANK SUM TEST IS CONDUCTED AT SIGNIFICANCE LEVEL $\alpha = 0.05$

F		DECC-G	DECC-DG	MLCC	DCC
DF1	Mean	1.4e-10	1.1e-05	5.4e-29	**0**
	Std	6.9e-10	1.6e-05	1.2e-28	0
	p	3.5e-08	3.5e-08	1.7e-01	-
DF2	Mean	1.0e+12	2.1e+11	5.6e+12	**6.5e+10**
	Std	3.9e+11	5.1e+10	2.0e+12	2.2e+10
	p	6.8e-08	6.8e-08	6.8e-08	-
DF3	Mean	2.5e+15	2.5e+15	2.5e+15	2.5e+15
	Std	1.2e+06	2.4e+10	5.1e+13	2.0e+06
	p	7.5e-09	1.0e-06	3.3e-06	-
DF4	Mean	3.9e+06	3.9e+06	3.9e+06	3.9e+06
	Std	1.3e+04	7.6e+03	1.7e+04	1.4e+03
	p	8.1e-02	1.0e-07	2.1e-06	-
DF5	Mean	7.9e+07	5.3e+07	3.1e+11	**5.4e+06**
	Std	1.6e+08	1.9e+08	7.1e+11	3.8e+06
	p	7.4e-05	1.3e-04	1.6e-07	-
DF6	Mean	5.4e+06	5.4e+06	5.4e+06	5.4e+06
	Std	9.8e-02	5.8e-09	1.4e-08	2.8e-08
	p	NaN	1.3e-04	1.6e-07	-
DF7	Mean	1.2e+13	1.3e+12	7.8e+13	**7.5e+11**
	Std	4.7e+29	6.2e+11	4.3e+13	3.7e+11
	p	6.8e-08	2.8e-03	6.8e-08	-
DF8	Mean	1.2e+16	1.2e+16	1.2e+16	1.2e+16
	Std	2.3e+07	2.5e+12	4.3e+12	4.6e+08
	p	NaN	8.0e-02	2.6e-05	-
DF9	Mean	6.9e+07	7.0e+07	6.9e+07	6.9e+07
	Std	5.3e+05	2.1e+05	5.4e+05	3.9e+05
	p	3.6e-02	9.2e-08	2.0e-01	-
DF10	Mean	3.7e+11	3.7e+11	2.3e+13	3.7e+11
	Std	4.1e+09	4.9e+09	5.2e+13	3.5e+09
	p	2.3e-03	1.8e-05	6.8e-08	-
OB12	Mean	2.9e+03	5.4e+06	1.1e+04	**1.8e+03**
	Std	2.7e+02	8.4e+06	4.1e+04	2.2e+02
	p	6.7e-08	6.8e-08	2.9e-01	-
OB13	Mean	3.3e+09	3.1e+09	2.9e+10	**2.1e+07**
	Std	1.3e+09	5.0e+09	1.1e+10	1.6e+07
	p	6.8e-08	6.8e-08	6.8e-08	-
OB14	Mean	3.78e+10	5.8e+10	5.5e+11	**6.1e+07**
	Std	4.62+10	1.1e+11	2.0e+11	1.6e+08
	p	6.8e-08	1.2e-07	6.8e-08	-
OA19	Mean	1.6e+06	**1.1e+06**	1.8e+06	1.6e+06
	Std	7.6e+05	7.5e+04	1.9e+05	1.4e+05
	p	2.8e-01	6.8e-08	6.8e-08	-
OA20	Mean	2.4e+03	3.2e+09	2.0e+03	**1.7e+03**
	Std	1.9e+02	3.3e+09	3.0e+03	3.3e+02
	p	2.3e-06	6.8e-08	1.1e-03	-
OB15	Mean	1.1e+07	9.2e+06	1.9e+11	**2.7e+06**
	Std	3.7e+06	4.0e+06	5.2e+11	1.5e+06
	p	7.9e-08	6.8e-08	2.0e-02	-
w/t/l	-	9/7/0	9/6/1	10/6/0	-

of deterministic subcomponents using greedy strategy. For overlapping functions, the subcomponents are supposed to be dynamically changed as evolutionary optimization process. Although DG-based decomposition methods take the interaction information into consideration, it is still not appropriate to divide all variables into a static set containing numerous deterministic subcomponents.

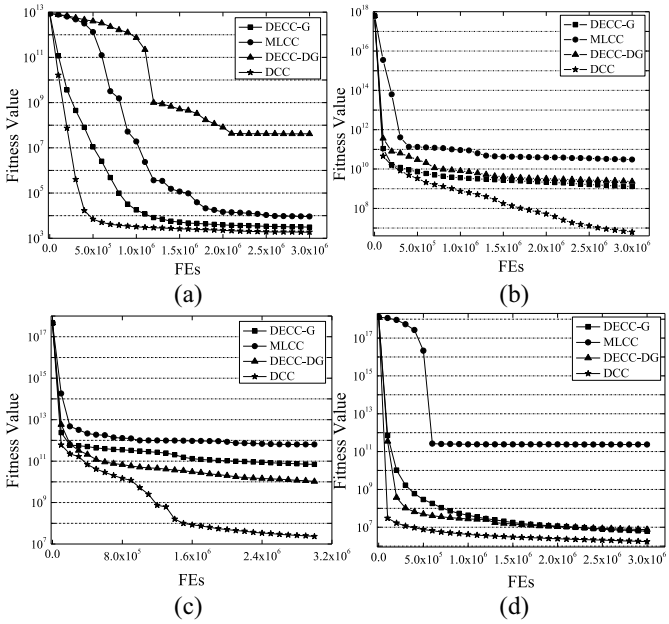

Fig. 5. Convergence curves of DCC, DECC-G, DECC-DG, and MLCC for functions OB12, OB13, OB14, and OB15.

Fig. 5(a)–(d) shows the convergence curves of DECC-G, DECC-DG, MLCC, and DCC on functions OB12, OB13, OB14, and OB15, where *x*-axis represent the FEs and *y*-axis represent the best fitness value. DCC achieves higher convergence rate than the other compared algorithms. Take Fig. 6(a) as an example, DCC obtains a solution of high quality at about 5×10^5 functions evaluations, whilst DECC-G, MLCC obtain the similar value at about 1.5×10^6 and 2×10^6 functions evaluations, respectively. DECC-DG is tracked into local optima at about 2×10^6 functions evaluations. For the other convergence curves, DCC still have the potential to achieve better solutions at the end of the evolutionary process due to its high gradient. Observing from Fig. 5(a)–(d), the performance of DECC-DG is close to those random-based CC frameworks. This phenomenon indicates that interaction relationship is not a dominate attribute of designing CC framework for large scale nonseparable problems. DCC considers the contribution of each variable, and the interaction relationship. The DyG method is capable of constructing the most superior subcomponent and allocating computational resources to the superior subcomponent. DCC works in a more computationally efficient manner than the non-CBCC frameworks. The experimental results ascertain the effectiveness of the dynamic mechanism over static or random methods.

C. Comparison With Contribution-Based CC Framework

In this section, experiments are conducted in order to investigate the performance of DCC in comparison with CBCC frameworks. Two typical CBCC frameworks (CBCC and CCFR) are adopted here. Considering the similar reason as the previous section, DG is used in CBCC and CCFR.

Experimental results of CBCC-DG, CCFR-DG, and DCC are summarized in Table III. As can be noted from Table III, DCC significantly outperforms the other algorithms on almost

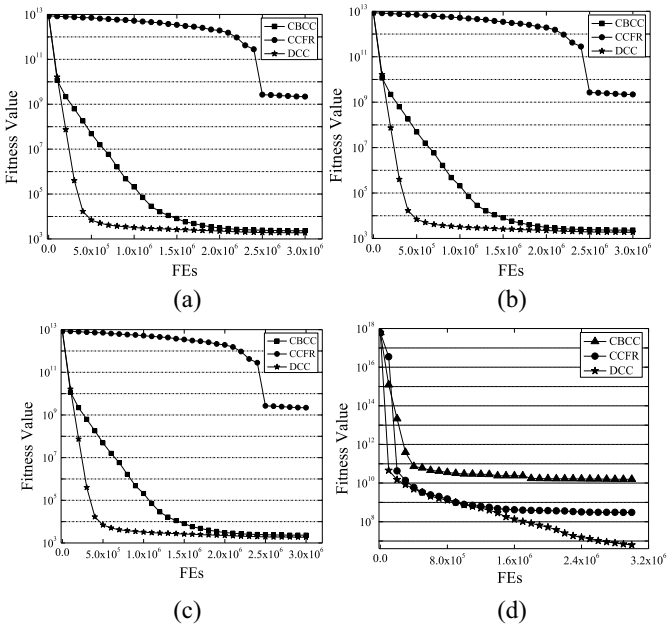

Fig. 6. Convergence curves of DCC, CBCC, CCFR, for functions OB12, OB13, OB14, and OB15.

all benchmark functions with several order of magnitude. As can be noted from the experimental results given in Tables II and III, the contribution-based algorithms show close or even inferior performance on nonseparable LSGO functions than non-CBCC frameworks on functions DF02, DF03, and OB13. According to the literatures of CBCC frameworks, their performances are influenced by the grouping accuracy. The overlapping and fully nonseparable LSGO functions are theoretically nonseparable from the perspective of interaction. Although the nonseparable LSGO functions can be partitioned using greedy strategy, e.g., the DG method, the grouping results are not accurate and appropriate. This explains why classic CBCC frameworks are not appropriate for solving overlapping LSGO problems. Nevertheless, it is still a great inspiration for computational resource allocation. In the proposed DCC framework, the nonseparable problems become separable. Owing to the novel DyG method, a series of dynamic subcomponents are constructed according to the dynamic contribution of variables and their interactive relationship. Computational resources can be dynamically allocated to those superior subcomponents. The better performance of the DCC framework reveals that the grouping results of the DyG method is more appropriate than that of the DG method.

Fig. 6(a)–(d) shows the convergence curves of CBCC-DG, CCFR-DG, and DCC on OB12, OB13, OB14, and OB15, respectively. In Fig. 7(a)–(d), DCC achieves higher convergence rate than the other compared algorithms. In Fig. 6(a), DCC achieves solutions of high quality at about 5×10^6 function evaluations, whilst CBCC-DG obtains a similar solution quality at about 1.6×10^6 function evaluations. CCFR-DG is trapped into local optima at about 1.9×10^6 function evaluations. For CBCC-DG and CCFR, they achieve close performance in terms of convergence rate. However, CCFR

TABLE III
EXPERIMENT ON 12 OVERLAPPING FUNCTIONS AND 3 NONSEPARABLE FUNCTIONS IN COMPARISON WITH NONCONTRIBUTION-BASED ALGORITHMS. ALL THE ALGORITHMS ARE CONDUCTED IN 30 INDEPENDENT TRIALS. THE AVERAGE AND STANDARD DEVIATION VALUES OF THE RESULTS ARE REPRESENTED BY MEAN, AND STD., RESPECTIVELY. THE HIGHLIGHTED ENTRIES ARE SIGNIFICANTLY BETTER. WILCOXON RANK SUM TEST IS CONDUCTED AT SIGNIFICANCE LEVEL $\alpha = 0.05$

F		CBCC-DG	CCFR-DG	DCC
DF01	Mean(Std)	1.4e+02(7.7e+02)	4.0e-04(5.5e-04)	**0**(0)
	p	3.5e-08	3.5e-08	-
DF02	Mean(Std)	3.6e+12(1.1e+12)	3.5e-08 (2.6e+13)	**6.5e+10**(2.2e+10)
	p	6.8e-08	6.8e-08	-
DF03	Mean(Std)	**2.5e+15**(5.2e+06)	2.6e+15(2.6e+13)	**2.5e+15**(2.0e+06)
	p	6.8e-08	8.0e-09	-
DF04	Mean(Std)	**4.0e+06**(8.3e+03)	**4.0e+06**(9.9e+03)	**4.0e+06**(1.4e+03)
	p	6.8e-08	6.8e-08	-
DF05	Mean(Std)	7.2e+08(2.2e+09)	5.8e+06(7.2e+06)	**5.4e+06**(3.8e+06)
	p	1.1e-07	7.6e-01	-
DF06	Mean(Std)	**5.4e+06**(5.3e-03)	**5.4e+06**(1.1e-05)	**5.4e+06**(2.8e-08)
	p	1.1e-07	7.6e-01	-
DF07	Mean(Std)	7.2e+13(1.4e+13)	**1.6e+11(2.0e+11)**	7.5e+11(3.7e+11)
	p	6.8e-08	9.1e-07	-
DF08	Mean(Std)	**1.2e+16**(3.0e+07)	**1.2e+16**(3.5e+11)	**1.2e+16**(4.6e+08)
	p	6.8e-08	7.5e-09	-
DF09	Mean(Std)	7.0e+07(1.4e+05)	7.1e+07(2.0e+05)	**6.9e+07**(3.9e+05)
	p	6.8e-08	6.8e-08	-
DF10	Mean(Std)	3.9e+11(9.9e+09)	3.7e+11(3.9e+09)	**3.6e+11(3.5e+09)**
	p	1.7e-07	1.2e-03	-
OB12	Mean(Std)	3.5e+3(9.3e+03)	2.1e+09(1.4e+08)	**1.8e+03(2.2e+02)**
	p	5.8e-01	6.8e-08	-
OB13	Mean(Std)	1.9e+10(3.7e+09)	6.8e+08(2.6e+09)	**2.1e+07**(1.6e+07)
	p	6.8e-08	2.1e-06	-
OB14	Mean(Std)	2.4e+11(6.1e+10)	5.1e+10(1.6e+11)	**6.1e+07**(1.6e+08)
	p	6.8e-08	7.4e-05	-
OA19	Mean(Std)	2.5e+06(1.7e+05)	**1.1e+06**(6.8e+04)	1.6e+06(1.4e+05)
	p	6.8e-08	6.8e-08	-
OA20	Mean(Std)	8.0e+08(1.1e+09)	3.6e+09(2.7e+09)	**1.7e+03**(3.3e+02)
	p	6.8e-08	6.8e-08	-
OB15	Mean(Std)	2.1e+07(5.2e+06)	1.4e+07(1.7e+07)	**2.7e+06(1.5e+06)**
	p	6.8e-08	6.8e-08	-
w/t/l	-	10/6/0	10/4/2	-

achieves significantly better performance than CBCC on partially separable LSGO problems according to [45]. This phenomenon indicates that the decomposition strategy becomes the bottleneck of CBCC frameworks on solving nonseparable LSGO problems. Owing to the dynamic decomposition mechanism, DCC achieves a significantly better performance than the compared algorithms.

D. Experiment on Fully Separable Functions

In this section, we investigate the performance of DCC on fully separable functions in comparison with DECC-G, MLCC, DECC-DG, CBCC-I, and CCFR-I. CBCC-I and CCFR-I represent that the corresponding CC frameworks incorporate ideal grouping. Fully separable function is a special type function, where variables are independent of each other. In an ideal grouping method, each variable of fully separable function is an independent subcomponent, and therefore the contribution of each variable can be accurately calculated. This ideal decomposition satisfies the requirement of contribution-based mechanism, and thus it is adopted in this

section to investigate the performance of DCC in terms of the CRA.

Experimental results are given in Appendix A. DCC achieves the best performance in the majority of the LSGO functions. For functions OA01, OA03, and OB01, DCC outperforms the other algorithms by several magnitudes. For random grouping-based algorithms (DECC-G and MLCC), variables are randomly divided into a number of subcomponents. Although each variable obtains the chance to undergo evolutionary optimization process, it is a way without any guideline and therefore inefficient. For DECC-DG, the subcomponents are optimized in a round-robin manner, which is not a computationally efficient way. Furthermore, the DG method divides a D-dimensional fully separable LSGO problem into D subcomponent, i.e., a variable represents a subcomponent. Although the contribution-based algorithms work in the computationally efficient way, the size of each subcomponent is too small to limit the ability of optimizer, which is a waste of the computational resources. For the fully separable LSGO problems, variables are independent from each other. The contribution information becomes the only criterion for the decomposition strategy of the DyG method. The computational resources are allocated to the superior variables with high contributions, which makes the proposed DCC work in a computationally efficient manner. Compared with static CC and CBCC frameworks, the way of CRA is more efficient in the proposed DCC framework. The experimental results indicate that the dynamic decomposition mechanism of DCC works well when dealing with fully separable objective functions.

E. Investigation on the Influence of Different PA Strategies

In this section, we investigate the influence of different PA strategies on the performance of CC frameworks. This is a first attempt to clarify the usage of various PA methods by experimental statistics. Two optimizers, i.e., JaDE and SaNSDE, are assembled with DCC and DECC-DG. Furthermore, fixed parameters setting are adopted for JaDE and SaNSDE.

Experimental results are listed in Appendix A. DCC-JA and DCC-JF represents that DCC is assembled with JaDE optimizer with self-adaptive parameters and fix parameters, respectively. Similarly, DCC-SA and DCC-JF represents DCC is assembled with JaDE optimizer with self-adaptive parameters and fix parameters, respectively. As can be noted from experimental results, optimizers with self-adaptive parameter strategy perform better than those with fixed parameter strategy for most of the functions, and the differences are significant. The experimental results indicate the adaptation mechanism of small scale optimization still works in large scale optimization. However, the performance of CC frameworks is influenced by different PA strategies. For functions OB01, OB02, OB12, OB13, OB14, and OB15, DCC with JaDE optimizer performs better than that with SaNSDE due to the high adaptation frequency when both of them are working in the SSPA mode.

V. Conclusion

This paper proposes a dynamic cooperative coevolutionary framework in order to solve nonseparable LSGO problems. DCC consists of three crucial parts: 1) the EDC; 2) the DyG strategy; and 3) the SSPA strategy. EDC evaluates the contribution of each dimension, which expands the usage of contribution-based mechanism from partially separable LSGO problems to nonseparable LSGO problems. DyG is totally different from the existing grouping strategies, which appropriately incorporates two critical attributes of the variables, i.e., interaction relationship and contribution, in solving the nonseparable LSGO problems. SSPA strategy is introduced to enhance the search efficiency of the optimizer.

Four types of experiments have been conducted to verify the effectiveness of DCC. The experimental results indicate that DCC achieves significantly superior performance in terms of solution quality and convergence rate over existing CC frameworks. For the optimization of the large scale nonseparable problems, it is more reasonable that the interaction attribute of variables is supposed to integrate with other attributes, e.g., the contribution, to achieve dimension reduction. The attributes of variables can be dynamically changed during the optimization process, and the DyG is a promising way of coping with this issue. Future work contains the smart use of different optimizers and the design of more effective parameter adaptation strategies.

Appendix A
Supplementary Material Available on the Web

The experiments in the supplementary material consist of the following parts.

1) The sensitivity test of the parameters (G_r, G_d, N_1, and N_2) of the proposed DCC.
2) Ablation study of the proposed DCC.
3) The summary experimental results of the compared algorithms.

Appendix B
Frequently Used Acronyms

Acronyms	*Description*
LSGO	Large scale optimization.
CC	Cooperative coevolution.
DCC	Dynamic cooperative coevolution
DG	Differential grouping.
CBCC	Contribution-based CC.
DyG	Dynamic grouping.
EDC	Estimation of the dimensional contribution.
SSPA	Stage-by-stage parameter adaptation.
SaNSDE	Self-adaptive differential evolution with neighborhood search.
FEs	Fitness evaluations.

References

[1] S. K. Goh, K. C. Tan, A. Al-Mamun, and H. A. Abbass, "Evolutionary big optimization (BigOpt) of signals," in *Proc. IEEE Congr. Evol. Comput.*, Sendai, Japan, 2015, pp. 3332–3339.

[2] M. S. Sorkherizi and A. A. Kishk, "Use of group delay of sub-circuits in optimization of wideband large-scale bandpass filters and diplexers," *IEEE Trans. Microw. Theory Techn.*, vol. 65, no. 8, pp. 2893–2905, Aug. 2017.

[3] V. Miranda, J. D. H. Martins, and V. Palma, "Optimizing large scale problems with metaheuristics in a reduced space mapped by autoencoders—Application to the wind-hydro coordination," *IEEE Trans. Power Syst.*, vol. 29, no. 6, pp. 3078–3085, Nov. 2014.

[4] Y. Cao and D. F. Sun, "A parallel computing framework for large-scale air traffic flow optimization," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1855–1864, Dec. 2012.

[5] P. Rodríguez-Mier, M. Mucientes, and M. Lama, "Hybrid optimization algorithm for large-scale QoS-aware service composition," *IEEE Trans. Services Comput.*, vol. 10, no. 4, pp. 547–559, Jul./Aug. 2017.

[6] I. Fister *et al.*, "Artificial neural network regression as a local search heuristic for ensemble strategies in differential evolution," *Nonlin. Dyn.*, vol. 84, no. 2, pp. 895–914, Apr. 2016.

[7] I. Fister, Jr., M. Perc, S. M. Kamal, and I. Fister, "A review of chaos-based firefly algorithms: Perspectives and research challenges," *Appl. Math. Comput.*, vol. 252, no. 1, pp. 155–165, Feb. 2015.

[8] B. Xue, M. J. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 606–626, Aug. 2016.

[9] D. B. Zhao, Y. J. Dai, and Z. Zhang, "Computational intelligence in urban traffic signal control: A survey," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 4, pp. 485–494, Jul. 2012.

[10] M.-H. Tayarani-Najaran, X. Yao, and H. M. Xu, "Meta-heuristic algorithms in car engine design: A literature survey," *IEEE Trans. Evol. Comput.*, vol. 19, no. 5, pp. 609–629, Oct. 2015.

[11] M. Z. Ali, N. H. Awad, P. N. Suganthan, and R. G. Reynoldss, "An adaptive multipopulation differential evolution with dynamic population reduction," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2768–2779, Sep. 2017.

[12] Y.-J. Gong *et al.*, "Genetic learning particle swarm optimization," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2277–2290, Oct. 2016.

[13] S. Strasser, J. Sheppard, N. Fortier, and R. Goodman, "Factored evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 281–293, Apr. 2017.

[14] J. C. Fan, J. Wang, and M. Han, "Cooperative coevolution for large-scale optimization based on kernel fuzzy clustering and variable trust region methods," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 4, pp. 829–839, Aug. 2014.

[15] R. Cheng and Y. C. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 191–204, Feb. 2015.

[16] Q. Yang *et al.*, "Segment-based predominant learning swarm optimizer for large-scale optimization," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2896–2910, Sep. 2017.

[17] Y.-F. Zhang and H.-D. Chiang, "A novel consensus-based particle swarm optimization-assisted trust-tech methodology for large-scale global optimization," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2717–2729, Sep. 2017.

[18] X.-Y. Zhang *et al.*, "Kuhn–Munkres parallel genetic algorithm for the set cover problem and its application to large-scale wireless sensor networks," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 695–710, Oct. 2016.

[19] Y. Pan, R. K. Xia, J. Yin, and N. Liu, "A divide-and-conquer method for scalable robust multitask learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 12, pp. 3163–3175, Dec. 2015.

[20] J. L. Zhang, C. Y. Zhang, T. G. Chu, and M. Perc, "Resolution of the stochastic strategy spatial Prisoner's dilemma by means of particle swarm optimization," *PLoS ONE*, vol. 6, no. 7, pp. 1–7, Jul. 2011.

[21] I. Fister, Jr., I. Fister, and M. Perc, "Toward the discovery of citation cartels in citation networks," *Front. Phys.*, vol. 4, pp. 1–49, Dec. 2016.

[22] M. A. Potter and K. A. D. Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving From Nature*. Heidelberg, Germany: Springer, 1994, pp. 249–257.

[23] H. Ge, L. Sun, G. Z. Tan, Z. Chen, and C. L. P. Chen, "Cooperative hierarchical PSO with two stage variable interaction reconstruction for large scale optimization," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2809–2823, Sep. 2017.

[24] M. N. Omidvar, X. D. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 378–393, Jun. 2014.

[25] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *Proc. IEEE Congr. Evol. Comput.*, Hong Kong, 2008, pp. 1110–1116.

[26] A. Song, W. N. Chen, P. T. Luo, Y. J. Gong, and J. Zhang, "Overlapped cooperative coevolution for large scale optimization," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 2017, pp. 3689–3694.

[27] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, "Benchmark functions for the CEC'2010 special session and competition on large-scale global optimization," Nat. Inspired Comput. Appl. Lab., Univ. Sci. Technol. China, Hefei, China, Rep., 2009.

[28] X. Li, K. Tang, M. N. Omidvar, Z. Yang, and K. Qin, "Benchmark functions for the CEC'2013 special session and competition on large-scale global optimization," School Comput. Sci. Inf. Technol., RMIT Univ., Melbourne, VIC, Australia, Rep., 2013.

[29] Z. Yang, K. Tang, and X. Yao, "Differential evolution for high-dimensional function optimization," in *Proc. IEEE Congr. Evol. Comput.*, Singapore, 2007, pp. 3523–3530.

[30] M. Perc *et al.*, "Statistical physics of human cooperation," *Phys. Rep.*, vol. 687, pp. 1–51, May 2017.

[31] Z. Wang *et al.*, "Statistical physics of vaccination," *Phys. Rep.*, vol. 664, no. 9, pp. 1–113, Dec. 2016.

[32] D. Helbing *et al.*, "Saving human lives: What complexity science and information systems can contribute," *J. Stat. Phys.*, vol. 158, no. 3, pp. 735–781, Feb. 2015.

[33] M. Perc and A. Szolnoki, "Coevolutionary games—A mini review," *Biosystems*, vol. 99, no. 2, pp. 109–125, Feb. 2010.

[34] M. A. Potter and K. A. D. Jong, "A cooperative coevolutionary approach to function optimization," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 1994, pp. 249–257.

[35] F. V. D. Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.

[36] W. X. Chen, T. Wise, Z. Y. Yang, and K. Tang, "Large-scale global optimization using cooperative coevolution with variable interaction learning," in *Proc. Int. Conf. Parallel Problem Solving Nat. II*, Kraków, Poland, 2010, pp. 300–309.

[37] Y. Sun, M. Kirley, and S. K. Halgamuge, "Extended differential grouping for large scale global optimization with direct and indirect variable interactions," in *Proc. Conf. Genet. Evol. Comput.*, Madrid, Spain, 2015, pp. 313–320.

[38] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *ACM Trans. Math. Softw.*, vol. 42, no. 2, Jun. 2016, Art. no. 13.

[39] M. N. Omidvar, M. Yang, Y. Mei, X. D. Li, and X. Yao, "DG2: A faster and more accurate differential grouping for large-scale black-box optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 6, pp. 929–942, Dec. 2017.

[40] Z. Y. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Inf. Sci.*, vol. 178, no. 15, pp. 2985–2999, Aug. 2008.

[41] X. D. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 2, pp. 210–224, Apr. 2012.

[42] Z. Y. Yang, K. Tang, and X. Yao, "Multilevel cooperative coevolution for large scale optimization," in *Proc. IEEE Congr. Evol. Comput.*, Hong Kong, 2008, pp. 1663–1670.

[43] M. N. Omidvar, X. D. Li, and X. Yao, "Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms," in *Proc. Genet. Evol. Comput. Conf.*, Dublin, Ireland, 2011, pp. 1115–1122.

[44] M. N. Omidvar, B. Kazimipour, X. D. Li, and X. Yao, "CBCC3—A contribution-based cooperative co-evolutionary algorithm with improved exploration/exploitation balance," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2016, pp. 3541–3548.

[45] M. Yang *et al.*, "Efficient resource allocation in cooperative co-evolution for large-scale global optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 493–505, Aug. 2017.

[46] J. Q. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.

[47] A. Ranganathan, "The Levenberg–Marquardt algorithm," *Tuts. LM Algorithm*, vol. 11, no. 1, pp. 101–110, 2004.

[48] J. M. Gablonsky and C. T. Kelley, "A locally-biased form of the DIRECT algorithm," *J. Glob. Optim.*, vol. 21, no. 1, pp. 27–37, Sep. 2001.

[49] B. Kazimipour, X. D. Li, and A. K. Qin, "Effects of population initialization on differential evolution for large scale optimization," in *Proc. IEEE Congr. Evol. Comput.*, Beijing, China, 2014, pp. 2404–2411.

Xin-Yuan Zhang (S'14) received the B.S. degree from Sun Yat-sen University, Guangzhou, China, in 2014, where he is currently pursuing the Ph.D. degree.

Since 2017, he has been a Research Assistant with the South China University of Technology, Guangzhou. His current research interests include evolutionary computation algorithms, swarm intelligence algorithms, large-scale optimization their applications in real-world problems, and smart grid.

Yue-Jiao Gong (S'10–M'15) received the B.S. and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2010 and 2014, respectively.

From 2015 to 2016, she was a Post-Doctoral Research Fellow with the Department of Computer and Information Science, University of Macau, Macau, China. She is currently a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. Her current research interests include evolutionary computation, swarm intelligence and their applications to intelligent transportation scheduling. She has published over 70 papers, including over 20 IEEE TRANSACTIONS papers in the above areas.

Ying Lin (M'13) received the Ph.D. degree in computer science from Sun Yat-sen University, Guangzhou, China, in 2012.

She is currently an Assistant Professor with the Department of Psychology, Sun Yat-sen University. Her current research interests include computational intelligence and its applications in psychometrics and neuroimaging.

Jie Zhang (M'99) received the master's degree in computer science from the China University of Mining and Technology, Xuzhou, China, in 1999.

She is currently an Associate Professor with the Beijing University of Chemical Technology, Beijing, China. Her current research interests include formal verification, and PHM for power and embedded system design.

Ms. Zhang is a member of the Chinese Institute of Electronics Embedded Expert Committee.

Sam Kwong (F'13) received the B.S. degree in electrical engineering from the State University of New York at Buffalo, Buffalo, NY, USA, in 1983, the M.S. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 1985, and the Ph.D. degree from the University of Hagen, Hagen, Germany, in 1996.

From 1985 to 1987, he was a Diagnostic Engineer with Control Data Canada, Mississauga, ON, Canada. He joined Bell Northern Research, Ottawa, ON, Canada, as a Scientific Staff Member. In 1990, he became a Lecturer with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, where he is currently a Professor with the Department of Computer Science. His current research interests include video and image coding and evolutionary algorithms.

Jun Zhang (M'02–SM'08–F'17) received the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Changjiang Chair Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. His current research interests include computational intelligence, cloud computing, data mining, and power electronic circuits. He has published over 200 technical papers in the above area.

Dr. Zhang was a recipient of the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011 and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, and the IEEE TRANSACTIONS ON CYBERNETICS.