Boosting Cooperative Coevolution for Large Scale Optimization With a Fine-Grained Computation Resource Allocation Strategy

Zhigang Ren, *Member, IEEE*, Yongsheng Liang, *Student Member, IEEE*, Aimin Zhang, *Member, IEEE*, Yang Yang, Zuren Feng, *Member, IEEE*, and Lin Wang, *Member, IEEE*

***Abstract*—Cooperative coevolution (CC) has shown great potential for solving large-scale optimization problems (LSOPs). However, traditional CC algorithms often waste part of the computation resource (CR) as they equally allocate CR among all subproblems. The recently developed contribution-based CC algorithms improve the traditional ones to a certain extent by adaptively allocating CR according to some heuristic rules. Different from existing works, this paper explicitly constructs a mathematical model for the CR allocation (CRA) problem in CC and proposes a novel fine-grained CRA (FCRA) strategy by fully considering both the theoretically optimal solution of the CRA model and the evolution characteristics of CC. FCRA takes a single iteration as a basic CRA unit and always selects the subproblem which is most likely to make the largest contribution to the total fitness improvement to undergo a new iteration, where the contribution of a subproblem at a new iteration is estimated according to its current contribution, current evolution status, as well as the estimation for its current contribution. We verified the efficiency of FCRA by combining it with the success-history-based adaptive differential evolution which is an excellent DE variant but has never been employed in the CC framework. Experimental results on two benchmark suites for LSOPs demonstrate that FCRA significantly outperforms existing CRA strategies and the resulting CC algorithm is highly competitive in solving LSOPs.**

***Index Terms*—Computation resource allocation (CRA), cooperative coevolution (CC), differential evolution (DE), large scale optimization.**

Manuscript received September 24, 2017; revised May 14, 2018; accepted July 10, 2018. Date of publication August 27, 2018; date of current version September 5, 2019. This work was supported in part by the National Nature Science Foundation of China under Grant 61873199 and Grant 61503300, in part by the China Post-Doctoral Science Foundation under Grant 2014M560784 and Grant 2016T90922, and in part by the Post-Doctoral Research Foundation of Shaanxi Province, China. This paper was recommended by Associate Editor Q. Zhang. *(Corresponding author: Zhigang Ren.)*

Z. Ren, Y. Liang, A. Zhang, and Y. Yang are with the School of Electronic and Information Engineering, Autocontrol Institute, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: renzg@xjtu.edu.cn; liangyongsheng@stu.xjtu.edu.cn; zhangam@xjtu.edu.cn; yyang@xjtu.edu.cn).

Z. Feng is with the State Key Laboratory for Manufacturing Systems Engineering, School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: fzr9910@xjtu.edu.cn).

L. Wang is with the School of Information Science and Technology, Northwest University, Xi'an 710127, China (e-mail: wanglin@nwu.edu.cn).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2018.2859635

I. Introduction

ACOMPANIED with the rapid development of big data techniques, there are more and more large-scale optimization problems (LSOPs) emerging in scientific research and engineering applications [1]–[3]. Due to the black-box characteristics of this kind of problem, the gradient-based mathematical programming methods are no longer applicable. The gradient-free evolutionary algorithms (EAs) still take effect, but lose their efficiency rapidly with the increase of the problem dimension. This is the so-called "curse of dimensionality" [4], the main reason for which consists in that the solution space of a problem grows exponentially with the increase of its dimension, and conventional EAs cannot adequately explore the solution space of an LSOP within acceptable computation time [5], [6].

By taking the idea of "divide-and-conquer," cooperative coevolution (CC) provides a natural way to solve LSOPs [7], [8]. It first decomposes an original LSOP into a set of smaller and simpler subproblems, then solves the LSOP by cooperatively optimizing all subproblems. There are mainly three algorithmic components in the CC framework, including the decomposition method, the optimizer for subproblems, and the computation resource allocation (CRA) strategy. The first two components attracted much research attention in recent years. As a result, a large number of decomposition methods were developed and almost all kinds of conventional EAs were introduced into the CC framework [9], [10]. By contrast, less research effort was put into the CRA strategy. The CRA problem refers to how to reasonably assign the available computation resource (CR) to all subproblems such that the original LSOP can be efficiently solved. Most existing CC algorithms follow the CRA strategy in the traditional CC which equally allocates CR among all subproblems by optimizing them in a round-robin fashion [5], [7], [8]. This straightforward scheme usually causes wasted CR, since the optimizer often stagnates on some subproblems before exhausting the CR assigned to them, while some other subproblems can make new contributions to the best overall solution if more CR is given.

To the best of our knowledge, by now five contribution-based CC (CBCC) frameworks, including CBCC1 CBCC2 [11], [12]; CBCC3 [13]; CC with adaptive optimizer iterations (CCAOI) [14]; and the CC framework named CCFR [15]; have been developed to overcome the limitations of the traditional CC. Instead of equal division, they adaptively assign CR to subproblems according to their contributions to the total fitness improvement. For example, CBCC1 and CBCC2 reward the subproblem by making the largest cumulative contribution with extra optimization chances; CBCC3 just allows the subproblem with the largest current contribution to attend a new cycle; CCFR tends to assign CR to the subproblem which achieves the largest average on the previous and current contributions.

These studies verified that contribution-based CRA really takes effect in improving the performance of CC. However, the CRA strategies they developed still show some obvious weaknesses. First, due to the absence of theoretical foundation, these strategies were all developed just according to some heuristic rules. Second, they all take hundreds of iterations as a basic CRA unit, which means that a subproblem must exhaust all of the iterations in a unit at each time it is selected for optimization. These kinds of coarse-grained CRA strategies reduce the interaction frequency among subproblems on the one hand, and may still waste some CR on the other hand. Finally, these strategies emphasize much on the contributions of subproblems, but consider little about their evolution status. In fact, a subproblem with a slightly smaller contribution is likely to contribute more at a new iteration if it stays in a more active evolution status.

To remedy the defects mentioned above, we make the following beneficial attempts in this paper.

1) Explicitly construct a mathematical model for the CRA problem in CC and deduce its optimal solution under a reasonable assumption.
2) Propose a fine-grained CRA (FCRA) strategy according to both the theoretically optimal solutions of the CRA model and the evolution characteristics of CC. Concretely, FCRA takes a single iteration instead of a large number of iterations as a basic CRA unit. When making an allocation decision, it fully considers the evolution status as well as the contributions of subproblems. This makes the allocation result approach the theoretically optimal one as much as possible.
3) Develop a novel CC framework called FCRACC, within which a new efficient CC algorithm is designed by integrating FCRA with the success-history-based adaptive differential evolution (SHADE) [16]. SHADE is an excellent differential evolution (DE) variant but has never been employed by CC.

The rest of this paper is organized as follows. Section II presents related work on CC. Section III describes the proposed FCRACC in detail, including the CRA model, the FCRA strategy, and the SHADE employed in FCRACC. Section IV reports experimental settings and results. Finally, conclusions are drawn in Section V.

II. Related Work

CC can effectively tackle an LSOP by cooperatively optimizing all of the lower-dimensional subproblems obtained through decomposition. It mainly contains three algorithmic components, that is, the decomposition method, the optimizer, and the CRA strategy. Next, we review them in sequence.

A. Decomposition Method

Decomposition plays a vital role in ensuring the performance of CC, and by now, a variety of decomposition methods has been developed. The original CC algorithms do not aim at solving LSOPs but improve the performance of conventional EAs on small- and medium-scale problems. Therefore, little effort was paid to the decomposition method and some simple and straightforward ones were employed. The earliest CC algorithm, that is, cooperative coevolutionary genetic algorithm (GA) proposed by Potter and De Jong [7], divides an n-dimensional problem into n 1-D subproblems. Liu *et al.* [5] and van den Bergh and Engelbrecht [17] employed the same decomposition method when scaling up evolutionary programming and particle swarm optimization (PSO) with CC, respectively. Potter and De Jong [8] also developed a splitting-into-half decomposition strategy for CC which was again used by Shi *et al.* [18] when designing the cooperative coevolutionary DE algorithm. More generally, van den Bergh and Engelbrecht [6] suggested partitioning an n-dimensional problem into k s-dimensional subproblems for some $ks = n$ and $s << n$, and integrated this decomposition method with PSO.

To summarize, these initial static decomposition methods can improve the performance of conventional EAs on certain kinds of problems, but they show limited or even a negative effect on LSOPs, especially on nonseparable ones [5]–[7]. To alleviate this issue, researchers developed random decomposition methods. Yang *et al.* [19] proposed the first random decomposition method called random grouping (RG). Different from static methods, which just perform once before optimization, RG stochastically divides all of the variables into k subcomponents of dimension s in every cycle of CC. To tackle the issue that the value of s is difficult to set, Yang *et al.* [20] further developed a multilevel CC algorithm. It selects an s value from a pool for the new cycle with a higher probability if this value achieves larger performance improvement in the last cycle. Omidvar *et al.* [21] showed that the probability of grouping all of the interacting variables into one subcomponent dramatically reduces with the increase of the number of interacting variables, and suggested increasing the grouping frequency by reducing the iteration times in a cycle.

In recent years, some learning-based decomposition methods were developed. They focus on making near optimal decomposition by detecting the interdependencies among variables. Weicker and Weicker [22] proposed the first variable interaction learning (VIL) method, the main principle behind which is that the fitness landscape related with a variable will change if its interacting variables change. Although this method is not exclusively introduced for LSOPs, its key idea makes obvious sense and is used by the subsequent VIL methods for LSOPs [23]–[25]. The main drawback of these VIL methods lies in that they require many samples for detecting an interdependency [26], [27]. Some other researchers suggested partitioning variables according to their correlations [28], [29]. However, the commonly used correlation coefficient is not a proper measure for the interaction of variables as two separable variables might be highly correlated with each other [30]. According to the observation that the variations of interacting variables in two consecutive optimization cycles are limited if they are grouped into different subcomponents, Omidvar *et al.* [30] proposed a decomposition method called delta grouping which divides a certain number of variables with similar variations into the same subcomponent. It was shown that delta grouping can outperform RG on a variety of LSOPs, but often loses its efficiency on the problems which have more than one group of interacting variables [30], [31]. To remedy this defect, Omidvar *et al.* [31] further proposed the differential grouping (DG) method. DG assigns two variables into the same subcomponent if one variable affects the fitness variation caused by the change of the other variable. To overcome the shortcoming of DG that it ignores indirect interdependencies among variables, Mei *et al.* [32] and Sun *et al.* [33] developed a global DG (GDG) and an extended DG (XDG), respectively, both of which explicitly detect the interdependency between each pair of variables. Much more recently, Hu *et al.* [34] proposed a fast interdependency identification method which further improves GDG and XDG by reducing their CR requirement. Omidvar *et al.* [35] also developed a new version of DG (called DG2) which reduces the CR requirement by reusing some samples and provides a more reliable detection threshold by estimating the magnitude of roundoff errors. Different from other learning-based decomposition methods, a mutual-information-based method was lately proposed by Sun *et al.* [36]. They consider two variables interdependent if the mutual information between one variable and the partial derivative *with respect to* the other variable is larger than a given threshold.

B. Optimizer

The optimizer in CC is used to solve the subproblems obtained by decomposition. So far, almost all kinds of conventional EAs have been introduced into CC [9], [10]. In their pioneering work on CC, Potter and De Jong [7] employed GA as an optimizer. So did the initial work on CC conducted by Weicker and Weicker [22] and Ray and Yao [28]. Potter and De Jong [8] further showed that CC can be taken as a general extension for any other EAs besides GA. Liu *et al.* [5] scaled up evolutionary programming with CC. Van den Bergh and Engelbrecht [6], [17]; Sun *et al.* [24]; Lin and Cheng [29]; and Li and Yao [37] casted the basic PSO into CC by employing different decomposition methods, while Ge *et al.* [27], Li and Yao [38], and Aote *et al.* [39] improved the basic PSO in different ways and took them as optimizers in their respective CC algorithms. Trunfio [14], Ren and Wu [40], and Zheng *et al.* [41] verified the effectiveness of the firefly algorithm, the artificial bee colony algorithm, and the biogeography-based optimization algorithm under the CC framework, respectively. Liu and Tang [42] introduced the covariance matrix adaptation evolution strategy (CMA-ES) [43] into CC, where the variables are partitioned according to their variances. Besides, Mei *et al.* [32] and Sun *et al.* [36] also employed CMA-ES as optimizers to verify the superiority of their decomposition methods. Among all kinds of EAs, DE was most widely used in CC. Shi *et al.* [18] first introduced the basic DE into CC. Afterward, an improved DE called the self-adaptive DE with neighborhood search (SaNSDE) [44] was frequently employed by Omidvar *et al.* [11], [13], [21], [30], [31], [35]; Yang *et al.* [15], [19], [20]; and Sun *et al.* [33] in their CC algorithms. When testing the efficiency of their decomposition methods, Chen *et al.* [23] and Hu *et al.* [34] selected JADE [45] and IDE [46] as optimizers, respectively. Besides, some other researchers developed new DE variants in their CC algorithms according to the characteristics of subproblems [3], [25].

C. CRA Strategy

Within the CC framework, the subproblems not only cooperate with each other by providing context vector for evaluating solutions [6], but also compete for limited CR. During the early research stage of CC, little attention was paid to the CRA problem and all subproblems were naturally assigned the same amount of CR.

Omidvar *et al.* [11] first indicated that there is usually an imbalance among the contributions of different subproblems to the total fitness and treating all of the subproblems equally often wastes part of CR. They further proposed two versions of CBCC, that is, CBCC1 and CBCC2, which divide each CC cycle into two phases: 1) exploration phase and 2) exploitation phase. During the exploration phase, all of the subproblems are optimized in a round-robin fashion and their contributions to the total fitness improvement are accumulated. Then, in the exploitation phase, the subproblem with the largest cumulative contribution is allowed to undergo extra optimization. The difference between CBCC1 and CBCC2 lies in that the former just allows the rewarded subproblem to undergo a single extra cycle, whereas the latter persistently optimizes the rewarded subproblem as long as it can still improve the fitness value. Kazimipour *et al.* [12] investigated the sensitivity of CBCC to the decomposition accuracy and the imbalance level, and revealed that CBCC1 is less sensitive to the two factors and can outperform the traditional CC in most cases.

Trunfio [14] pointed out that the performance of CBCC1 and CBCC2 is influenced too much by the historical information accumulated during the early evolution stages, and developed a CC variant called CCAOI. CCAOI still optimizes all of the subproblems in a round-robin fashion, but adaptively adjusts the iteration times of each subproblem in a new cycle according to its average contribution over all of the iterations in the current cycle. The larger the average contribution of a subproblem, the more iterations this subproblem will be rewarded. Omidvar *et al.* [13] later claimed that besides being influenced too much by the historical information, CBCC1 and CBCC2 also suffer from over-exploration and over-exploitation, respectively. The new version of CBCC, which is CBCC3, that they developed tries to achieve better balance between exploration and exploitation by selecting the subproblem which makes the largest current contribution

Algorithm 1: General CC

1 Generate a decomposition $\boldsymbol{x} \to \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k\}$;
2 Initialize the population of each subproblem i: $\boldsymbol{P}_i, i = 1, \ldots, k$;
3 Initialize the best overall solution $\boldsymbol{x}^*$;
4 **while** the termination condition is not met **do**
5   Determine the subproblem i to be optimized;
6     $(\boldsymbol{P}_i, \boldsymbol{x}^*) \leftarrow optimizer(\boldsymbol{P}_i, \boldsymbol{x}^*, FEs)$;
7 **return** $\boldsymbol{x}^*$. // FEs specifies allowed iteration times in a cycle.

attend a new cycle and occasionally perform exploration during the optimization process. Aiming at the defects of CBCC1 and CBCC2, Yang *et al.* [15] recently proposed a new CC framework named CCFR. CCFR shares some characteristics with CBCC3. The main difference between them lies in that CCFR explicitly excludes stagnant subproblems and quantifies the contribution of an active subproblem by averaging its current contribution and previous contribution. It was numerically shown that both CBCC3 and CCFR perform much better than CBCC1 and CBCC2.

III. Description of FCRACC

Many large-scale real-world optimization problems are separable, where partially additively separable problems are the most common ones and were extensively studied in the CC research field [31]–[35].

Definition 1: A function is partially additively separable if it has the following general form:

$$f(\boldsymbol{x}) = \sum_{i=1}^{k} f_i(\boldsymbol{x}_i) \tag{1}$$

where $\boldsymbol{x} = (x_1, \ldots, x_n)$ *is* a global decision vector of n dimensions, $\boldsymbol{x}_i$ are mutually exclusive decision vectors of $f_i(\cdot)$, and k is the number of independent subcomponents [31].

To alleviate the curse of dimensionality, CC tries to solve $f(\boldsymbol{x})$ by first identifying mutually exclusive subcomponents $\boldsymbol{x}_i (i = 1, \ldots, k)$ and then cooperatively optimizing the resulting lower-dimensional subproblems $f_i(\boldsymbol{x}_i)$. It is notable that the expression of each $f_i(\cdot)$ is unknown since $f(\cdot)$ is a black-box optimization problem in the context of CC. To evaluate the solutions of each subproblem, CC maintains a context vector which is generally set as the best overall solution obtained so far [6]. More concretely, CC evaluates a solution $\boldsymbol{x}_i$ of $f_i(\boldsymbol{x}_i)$ by means of $f(\boldsymbol{x}^*|\boldsymbol{x}_i)$, where $\boldsymbol{x}^*|\boldsymbol{x}_i$ denotes the overall solution that replaces the corresponding subcomponent of the best overall solution $\boldsymbol{x}^*$ with $\boldsymbol{x}_i$. $\boldsymbol{x}^*$ will be updated by $\boldsymbol{x}^*|\boldsymbol{x}_i$ if it is outperformed by the latter. Algorithm 1 shows a general framework of CC. For traditional CC, step 5 selects different subproblems in a round-robin fashion.

A. CRA Model and Its Solution

In the CC framework, the computation or simulation of $f(\boldsymbol{x})$ is the most time-consuming part. For the given decomposition result on an LSOP and the selected optimizer, the CRA problem refers to how to reasonably assign the allowed maximum number of fitness evaluations (FEs) to all of the subproblems such that the maximum total fitness improvement can be achieved. For the additively separable function defined by Definition 1, the total fitness improvement is equal to the sum of the improvements achieved on all subproblems. Then the CRA problem can be mathematically modeled as follows:

$$\begin{aligned} &\arg \max_{m_1,\ldots,m_k} \sum_{i=1}^{k} \bar{C}_i(m_i) \\ &\text{s.t.} \ \sum_{i=1}^{k} m_i \le m \\ &\qquad m_i \ge 0, \quad i = 1, \ldots, k \end{aligned} \tag{2}$$

where m, m_i, and $\bar{C}_i(\cdot)$ denote the allowed maximum number of FEs, the number of FEs assigned to the ith subproblem, and the expectation of the fitness improvement achieved on the ith subproblem, respectively. As reviewed in Section II, CC employs EA which has a certain randomness as an optimizer, then the contribution $C_i(\cdot)$ made by a subproblem i is a random function with respect to m_i; therefore, the expectation of $C_i(\cdot)$ is adopted in (2). It is comprehensible that $\bar{C}_i(m_i)$ is nondecreasing with the increase of m_i. However, now it is impossible to deduce the closed expression of $\bar{C}_i(\cdot)$ since it also strictly relates to the landscape characteristics of $f_i(\cdot)$ and the search property of the selected optimizer, both of which are hard to mathematically describe. This means that it is unrealistic to pursue the theoretically optimal solution of the CRA model described by (2), and it is necessary to simplify this model.

For all of the subproblems obtained by decomposition, most CC algorithms employ the same optimizer with the same population size. Based on this fact, the CRA model presented in (2) can be changed to the following form:

$$\begin{aligned} &\arg \max_{t_1,\ldots,t_k} \sum_{i=1}^{k} \bar{C}_i(t_i p) \\ &\text{s.t.} \ \sum_{i=1}^{k} t_i\, p \le m \\ &\qquad t_i \ge 0, \quad i = 1, \ldots, k \end{aligned} \tag{3}$$

where p and t_i denote the population size of the employed optimizer and the iteration times allocated to the ith subproblem, respectively. For this new CRA model, the following theorem provides the optimal solution under a reasonable assumption.

Theorem 1: Let $\Delta\bar{C}_i^j$ denote the expectation of the contribution made by the ith $(i = 1, \ldots, k)$ subproblem at the jth iteration allocated to it and assume $\Delta\bar{C}_i^1 \ge \Delta\bar{C}_i^2 \ge \ldots$, then the optimal solution to the CRA problem described by (3) is

$$t_i^* = |\Delta\vec{\boldsymbol{C}}_i^{\lfloor m/p \rfloor} \bigcap \Delta\vec{\boldsymbol{C}}^{\lfloor m/p \rfloor}|, \quad i = 1, \ldots, k \tag{4}$$

where $\lfloor\cdot\rfloor$ denotes the round down operation, and $\Delta\vec{\boldsymbol{C}}_i^{\lfloor m/p \rfloor}$ and $\Delta\vec{\boldsymbol{C}}^{\lfloor m/p \rfloor}$ are the sets composed of $\Delta\bar{C}_i^j (j = 1, \ldots, \lfloor m/p \rfloor)$ and the top $\lfloor m/p \rfloor$ largest $\Delta\bar{C}_i^j (i = 1, \ldots, k; j = 1, \ldots, \lfloor m/p \rfloor)$, respectively.

Proof: The total contribution of a subproblem is equal to the sum of its contributions made at all of the iterations allocated to it, then

$$\sum_{i=1}^{k} \bar{C}_i(t_i p) = \sum_{i=1}^{k} \sum_{j=1}^{t_i} \Delta\bar{C}_i^j. \tag{5}$$

For the given m and p, the optimizer is allowed to conduct a maximum of $\lfloor m/p \rfloor$ iterations in total. Considering the fact that $\Delta\bar{C}_i^j \geq 0$ and the assumption that $\Delta\bar{C}_i^1 \geq \Delta\bar{C}_i^2 \geq \dots$ for $\forall i = 1, \dots, k$ and $\forall j = 1, 2, \dots$, we can know that maximizing $\sum_{i=1}^{k} \bar{C}_i(t_i p)$ implies selecting the top $\lfloor m/p \rfloor$ largest $\Delta\bar{C}_i^j$ $(i = 1, \dots, k; j = 1, \dots, \lfloor m/p \rfloor)$, which means $t_i^* = |\Delta\overrightarrow{\boldsymbol{C}}_i^{\lfloor m/p \rfloor} \bigcap \Delta\overrightarrow{\boldsymbol{C}}^{\lfloor m/p \rfloor}|$.

Remark 1: The assumption that $\Delta\bar{C}_i^1 \geq \Delta\bar{C}_i^2 \geq \dots$ for $\forall i = 1, \dots, k$ makes sense since it is indeed more likely for the optimizer to make a larger fitness improvement during its earlier search stages on a given optimization problem.

B. FCRA Strategy

Equation (4) provides the theoretically optimal solution for the CRA problem that is of concern in CC. However, when applying this solution, it is impossible to calculate each $t_i^*(i = 1, \dots, k)$ before the optimization process since the expression of each $\Delta\bar{C}_i^j$ and the resulting contribution sets, $\Delta\overrightarrow{\boldsymbol{C}}_i^{\lfloor m/p \rfloor}$ and $\Delta\overrightarrow{\boldsymbol{C}}^{\lfloor m/p \rfloor}$, are unknown. On the other hand, if the optimizer could select a right subproblem at each iteration, the optimal CRA result will be naturally achieved. According to this idea, the FCRA strategy proposed in this paper tackles this issue in an iteration-wise way as follows: once the optimizer finishes its current iteration on a subproblem i, FCRA immediately estimates the expectation of the fitness improvement that the optimizer can achieve on this subproblem at the next iteration (denote this expectation and its estimate as $\Delta\bar{C}_i^{\mathrm{n}}$ and $\Delta\tilde{C}_i^{\mathrm{n}}$, respectively), and keeps the estimates for the other subproblems fixed. Then, FCRA selects the subproblem with the largest estimation value for real optimization at the next iteration.

Mathematically, FCRA initializes each $\Delta\tilde{C}_i^{\mathrm{n}}(i = 1, \dots, k)$ to zero and updates one of the current subproblems as follows:

$$\Delta C_i^{\mathrm{c}} = \max\left(\max_{j=1,\dots,p} \Delta F_{ij}^{\mathrm{c}}, 0\right) \tag{6}$$

$$\delta_i^{\mathrm{c}} = \mathrm{std}\left(\Delta F_{ij}^{\mathrm{c}}, j = 1, \dots, p\right) \tag{7}$$

$$\Delta\tilde{C}_i^{\mathrm{n}} \leftarrow \alpha \cdot \Delta\tilde{C}_i^{\mathrm{n}} + (1-\alpha)\left(\Delta C_i^{\mathrm{c}} + \delta_i^{\mathrm{c}}\right), i = 1, \dots, k \tag{8}$$

where $\Delta F_{ij}^{\mathrm{c}}$ represents the real fitness improvement made by the jth population individual of the ith subproblem at the current iteration; ΔC_i^{c} and δ_i^{c} represent the maximum and the standard deviation of $\Delta F_{ij}^{\mathrm{c}}$ for all $j = 1, \dots, p$, respectively; and α is a forgetting factor satisfying $0 \leq \alpha < 1$. ΔC_i^{c} reflects the current contribution of the ith subproblem to the total fitness improvement. It is very likely for the optimizer to achieve larger fitness improvement at a new iteration on the subproblem with larger ΔC_i^{c}. δ_i^{c} reflects the current evolution status of the optimizer on the ith subproblem. The larger δ_i^{c} is, the more active the optimizer is, and the easier it can get a better solution. The introduction of α smoothens the volatility of ΔC_i^{c} and δ_i^{c} by taking advantage of the previous estimation value of $\Delta\tilde{C}_i^{\mathrm{n}}$.

Algorithm 2: FCRACC

1 Generate a decomposition $\boldsymbol{x} \rightarrow \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_k\}$;
2 Initialize the best overall solution $\boldsymbol{x}^*$and the parameter α;
3 *isInitialization* $\leftarrow$ *True*, $i \leftarrow 0$;
4 **while** the termination condition is not met **do**
5   **if** *isInitialization* == *True* and $i < k$ **then**
6         $i \leftarrow i + 1$, $\Delta\tilde{C}_i^{\mathrm{n}} \leftarrow 0$;
7   Initialize the population of the ith subproblem: $\boldsymbol{P}_i \leftarrow \{\boldsymbol{x}_{i1}, \dots, \boldsymbol{x}_{ip}\}$;
8   **Else**
9         $i \leftarrow \arg\max_{i=1,\dots k} \Delta\tilde{C}_i^{\mathrm{n}}$, *isInitialization* $\leftarrow$ *False*;
10     $(\boldsymbol{P}_i, \boldsymbol{x}^*, \Delta C_i^{\mathrm{c}}, \delta_i^{\mathrm{c}}) \leftarrow$ *optimizer*$(\boldsymbol{P}_i, \boldsymbol{x}^*,$ *isInitialization*$)$;
11     $\Delta\tilde{C}_i^{\mathrm{n}} \leftarrow \alpha \cdot \Delta\tilde{C}_i^{\mathrm{n}} + (1-\alpha)(\Delta C_i^{\mathrm{c}} + \delta_i^{\mathrm{c}})$;
12 **return** $\boldsymbol{x}^*$.

Remark 2: Most EAs can get better solutions during their early search stages, then both ΔC_i^{c} and δ_i^{c} take effect in allocating CR in this case. However, they seldom get better solutions during their late search stages, then δ_i^{c} plays the main role in this case. If the optimizer stagnates on a subproblem i, which means $\Delta C_i^{\mathrm{c}} = 0$ and $\delta_i^{\mathrm{c}} = 0$, then this subproblem will be soon excluded from CRA candidates.

Remark 3: In fact, $\Delta\tilde{C}_i^{\mathrm{n}}$ can hardly approximate $\Delta\bar{C}_i^{\mathrm{n}}$ with high enough accuracy as the latter is affected by many factors, but it can usually distinguish the relative magnitudes of a pair of $\Delta\bar{C}_i^{\mathrm{n}}$, thus enabling FCRA to make right allocation decisions.

Remark 4: FCRA takes a single iteration involving p FEs as a basic allocation unit. This endows it with an ability to provide much more fine-grained allocation results than existing CRA strategies since they all take hundreds of iterations as a basic allocation unit. Profiting from the above advantages, FCRA can greatly enhance the utilization efficiency of available CR.

Integrating FCRA into CC, we obtain the framework of FCRACC shown in Algorithm 2. Step 2 initializes the best overall solution $\boldsymbol{x}^*$ with an arbitrary feasible solution. Steps 3–7 initialize the population of each subproblem and enable all subproblems to undergo an iteration in sequence during the first k iterations. When a subproblem is optimized in step 10, the corresponding ΔC_i^{c} and δ_i^{c} are calculated, which facilitates the update of $\Delta\tilde{C}_i^{\mathrm{n}}$ in step 11. After the first k iterations, step 9 selects the subproblem with the largest $\Delta\tilde{C}_i^{\mathrm{n}}$ to optimize according to the proposed FCRA strategy. Steps 4–11 will be repeated until the termination condition is met.

Compared with existing CC frameworks, FCRACC cancels the concept of cycle and systematically allocates CR among subproblems based on the optimal solution of an explicit CRA model. It allows a subproblem to undergo just a single iteration once it is selected for optimization, which facilitates achieving much more fine-grained allocation results on the one hand and increasing the interaction frequency among subproblems on the other hand. It has been shown that a high interaction frequency is beneficial to improve the performance of CC [47].

C. SHADE for FCRACC

SHADE is an excellent DE variant developed in recent years [16]. It inherits the efficient "current-to-*p*best/1" mutation operator from the classic JADE algorithm [45], but further improves JADE with a different parameter (including mutation factor and crossover rate) adaptation mechanism. Instead of employing a single pair of parameter means, SHADE maintains a diverse set of parameter means to guide parameter adaptation. In this way, the negative impact of some poor parameter values can be reduced. SHADE and its variants made great success in solving small- and medium-scale optimization problems [48], but have never been employed by CC. This paper makes the first attempt to scale up SHADE for LSOPs. Algorithm 3 presents the main process of SHADE used in FCRACC. For more details of SHADE, the readers can refer to [16]. Here, we just specify two points of changes that make SHADE adapt to FCRACC and simplify its implementation.

First, step 10 evaluates the solution of a subproblem according to its fitness improvement to the best overall solution $\boldsymbol{x}^*$ instead of its fitness itself. The main reason is that the selection operator in SHADE needs to compare each population individual against the corresponding trial vector to determine the survivor for the next iteration. However, each individual has been evaluated at the last iteration, when the context vector, that is $\boldsymbol{x}^*$, may be different from the current one. This makes it infeasible to identify a better solution from the two candidates by directly comparing their fitness values. A straightforward way is to re-evaluate each individual, but this requires more CR. According to the property of additive separability defined by (1), the fitness improvement made by the solution of a subproblem to $\boldsymbol{x}^*$ eliminates the influence caused by the solutions of other subproblems and, thus, provides the comparability between an individual and its trial vector. It is notable that steps 18–20 and 4–6 update the fitness improvements of all individuals when $\boldsymbol{x}^*$ is updated. The aim is to make individuals share the same context vector with their respective trial vectors at the next iteration. Besides SHADE, this kind of indirect solution evaluation method also applies to other DE variants as long as they take the same selection operator with SHADE.

Second, SHADE maintains an external archive to provide a solution for the current-to-*p*best/1 mutation operator. The original SHADE initializes it with an empty set and continually fills or updates it with population individuals which are worse than corresponding trial vectors. This asks SHADE to perform different operations according to the real number of solutions in the archive during the entire evolution process, although the archive will be definitely fully filled after several iterations. To simplify operations, step 3 directly initializes the archive with a specified number of random solutions. This minor change significantly simplifies the implementation of SHADE without affecting its performance.

IV. Experimental Studies

The purpose of this section is three-fold: 1) to study the influence of parameters; 2) to investigate the behavior of FCRACC; and 3) to evaluate its performance.

Algorithm 3: $(\boldsymbol{P}_i, \boldsymbol{x}^*, \Delta C_i, \delta_i) \leftarrow$ *SHADE*$(\boldsymbol{P}_i, \boldsymbol{x}^*,$ *isInitialization*$)$

1 **if** *isInitialization* == *True* **then**
2 Initialize the parameter memory M_i for the ith subproblem;
3 Initialize the external archive $\boldsymbol{A}_i$ with $|\boldsymbol{P}_i|$ random solutions;
4 $j^* \leftarrow \arg\min_{j=1,\ldots p} f(\boldsymbol{x}^*|\boldsymbol{x}_{ij}), \boldsymbol{x}^* \leftarrow \boldsymbol{x}^*|\boldsymbol{x}_{ij^*}$;
5 **for** each $\boldsymbol{x}_{ij} \in \boldsymbol{P}_i$ **do**
6   $\Delta F_{ij} \leftarrow f(\boldsymbol{x}^*) - f(\boldsymbol{x}^*|\boldsymbol{x}_{ij})$;
7 **for** *each* $\boldsymbol{x}_{ij} \in \boldsymbol{P}_i$ **do**
8 Generate a pair of control parameters based on M_i;
9 Generate a trial vector $\boldsymbol{u}_{ij}$ according to "current-to-*p*best/1/bin";
10 $\Delta F'_{ij} \leftarrow f(\boldsymbol{x}^*) - f(\boldsymbol{x}^*|\boldsymbol{u}_{ij})$; // Evaluate $\boldsymbol{u}_{ij}$
11 **if** $\Delta F'_{ij} > \Delta F_{ij}$ **then**
12   Update a randomly selected solution in $\boldsymbol{A}_i$ with $\boldsymbol{x}_{ij}$;
13   Record current control parameters;
14   $\boldsymbol{x}_{ij} \leftarrow \boldsymbol{u}_{ij}, \Delta F_{ij} \leftarrow \Delta F'_{ij}$;
15 Update M_i based on the recorded successful control parameters;
16 $\Delta C_i \leftarrow \max(\max(\Delta F_{ij}), 0), \Delta\delta_i \leftarrow \mathrm{std}(\Delta F_{ij})$;
17 **if** $\Delta C_i > 0$ **then**
18 $j^* \leftarrow \arg\max_{j=1,\ldots p} \Delta F_{ij}, \boldsymbol{x}^* \leftarrow \boldsymbol{x}^*|\boldsymbol{x}_{ij^*}$;
19 **for** each $\boldsymbol{x}_{ij} \in \boldsymbol{P}_i$ **do**
20   $\Delta F_{ij} \leftarrow f(\boldsymbol{x}^*) - f(\boldsymbol{x}^*|\boldsymbol{x}_{ij})$;
21 **return** $\boldsymbol{P}_i, \boldsymbol{x}^*, \Delta C_i, \delta_i$.

TABLE I
CLASSIFICATION OF BENCHMARK FUNCTIONS

Separability	CEC2010 suite	CEC2013 suite
Separable functions	f_1 - f_3	f_1 - f_3
Partially separable functions	f_4 - f_{18}	f_4 - f_{11}
Nonseparable functions	f_{19} - f_{20}	f_{12} - f_{15}

A. Experimental Settings

The CEC2010 and CEC2013 benchmark suites, which contain 20 and 15 LSOPs, respectively, were employed in our experiments. All of these benchmark functions are minimization problems of 1000-D except that f_{13} and f_{14} in CEC2013 suite have 905 variables. Table I presents the classification of these functions. For more details about them, readers can refer to [49] and [50].

To evaluate the performance of FCRACC, we compared it with seven well-known CC algorithms (including the traditional CC [7], CBCC1, CBCC2 [11], CCFR [15], DE with CC (DECC) and RG (DECC-G) [19], DECC with delta grouping (DECC-D) [30], and DECC with DG2 [35]) and two state-of-the-art non-CC algorithms (including the memetic algorithm that creates local search chains with Solis Wets' method (MA-SW-Chains) [51] and the multiple offspring sampling framework for CEC2013 competition (MOS-CEC2013) [52]). Two decomposition methods were used in our experiments. One is the ideal decomposition which groups variables according to the prior knowledge of a benchmark function. The other is the recently developed DG2 method [35]. It improves the classic DG method [31] and can decompose LSOPs with high accuracy. Since the seven CC algorithms listed above all employ SaNSDE [44] as an optimizer, we also implemented a SaNSDE version of FCRACC besides the SHADE version.

As a result, four FCRACC algorithms in total were implemented. For the convenience of description, we abbreviate the FCRACC algorithm which adopts ideal decomposition and SHADE to $\text{FCRACC}_{\text{ID+SHADE}}$. Similar abbreviations can be given to other CC algorithms. To make fair comparisons, the parameters of SaNSDE in FCRACC were set to the same values as those in the seven existing CC algorithms and the parameters of SHADE were set according to the recommendation of the original paper [16], which means the population sizes of SaNSDE and SHADE were set to 50 and 100, respectively.

As suggested by [49], a maximum number of 3.0×10^6 FEs was used as the termination condition of a run of an algorithm. Unless otherwise mentioned, the results of each algorithm on a function were calculated based on 25 independent runs.

B. Influence of Parameters

FCRACC only has a single free parameter, that is, the forgetting factor α in (8). It determines the utilization degree of the previous estimation value of a subproblem's contribution when predicting a new contribution value for it. The larger α is, the more previous information will be used, but the less information about the current contribution and evolution status will be exploited. It is expected that α can achieve a good balance between these two kinds of information.

To investigate the sensitivity of FCRACC to α, we tested $\text{FCRACC}_{\text{ID+SHADE}}$ on a variety of benchmark functions with different α values taken from {0, 0.1, 0.3, 0.5, 0.7, 0.9, 0.95, 0.98, 0.99, 1}. Taking the CEC2010 function f_{16} and the CEC2013 function f_8 as examples, Fig. 1 presents the average, smallest, and largest fitness values obtained by the algorithm. It can be observed that $\text{FCRACC}_{\text{ID+SHADE}}$ performs well when α is less than 1 and only shows performance deterioration when α is very close to 1. In fact, $\text{FCRACC}_{\text{ID+SHADE}}$ with $0 \leq \alpha < 1$ can get a near optimal solution for f_{16} in almost each run. This means that the performance of $\text{FCRACC}_{\text{ID+SHADE}}$ is rather robust to the change of α. Only when the previous estimation information is emphasized too much, will the performance of FCRACC deteriorate, since CR will be improperly allocated in this case. For this reason, we recommend selecting a value for α from the interval [0, 0.95]. It was set to 0.5 in the experiments described below.

C. Behavior of FCRACC

This section studies the behavior of FCRACC under the ideal decomposition. In this case, all FEs are used for optimization. Fig. 2 shows the CRA and optimization results of $\text{FCRACC}_{\text{ID+SHADE}}$ in a single run, where the CEC2010 function f_{16} is employed as an example. This function can be ideally decomposed into 20 subproblems of the same dimension and the same contribution weight [49]. Fig. 2(a) presents the indices of the subproblems optimized at the first 5000 iterations, from which it can be seen that the optimization process on f_{16} can be divided into three stages. In the first stage, each subproblem is occasionally optimized at some iterations after all subproblems are sequentially optimized once within the first 20 iterations. The second stage starts from about the 100th

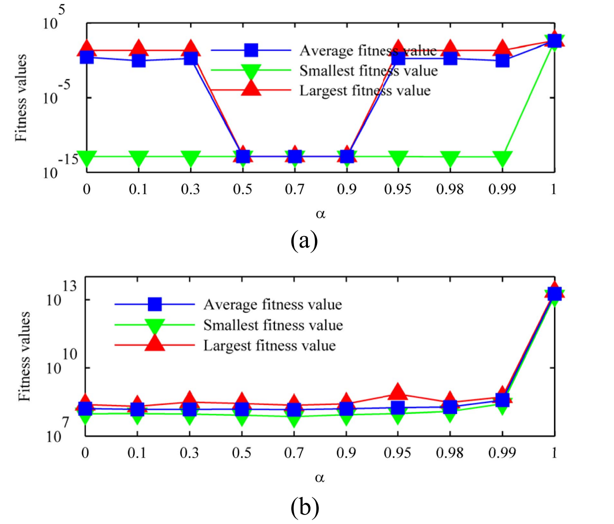

Fig. 1. Influence of α on the performance of FCRACC. (a) CEC2010 function f_{16}. (b) CEC2013 function f_8.

iteration and ends at about the 3400th iteration, during which each subproblem is deeply optimized within a certain number of consecutive iterations. After entering the third stage, the optimizer frequently changes its optimization object again.

The above phenomenon can be illustrated by Fig. 2(b) which presents the largest and the second largest contribution values estimated by $\text{FCRACC}_{\text{ID+SHADE}}$ at each iteration and the corresponding smallest overall fitness value it obtains. In the first stage, the search pattern of the optimizer on each subproblem is so random that few better solutions can be found, then the contribution made by the current subproblem tends to decrease. As a result, it is much likely to be replaced by a new subproblem with the largest contribution. In the second stage, the optimizer explores and exploits good solution regions of each subproblem. When a subproblem is optimized in this stage, its contribution first increases and then decreases, since the optimizer necessarily becomes more and more inactive and can achieve few improvements after its success in the explored good solution regions. All subproblems share similar status after they are deeply optimized once in the second stage, and no significant improvement can be achieved on each of them. Accordingly, the differences among their contributions are small, which leads to the frequent change of the optimization object in the third stage.

Fig. 2(c) presents the allocation result of a total of 3.0×10^4 iterations among 20 subproblems versus the number of iterations at which the optimizer gets better solutions for a subproblem, and Fig. 2(d) presents the given contribution weight of each subproblem versus the real fitness improvement achieved on it. It can be seen that the four performance indicators are rather consistent on all subproblems except that the tenth subproblem is assigned with many more iterations. The reason is that the optimizer finds a better solution for this subproblem in the late search process, when it stagnates on all other subproblems. As a result, the optimizer persistently optimizes this subproblem until also stagnating on it.

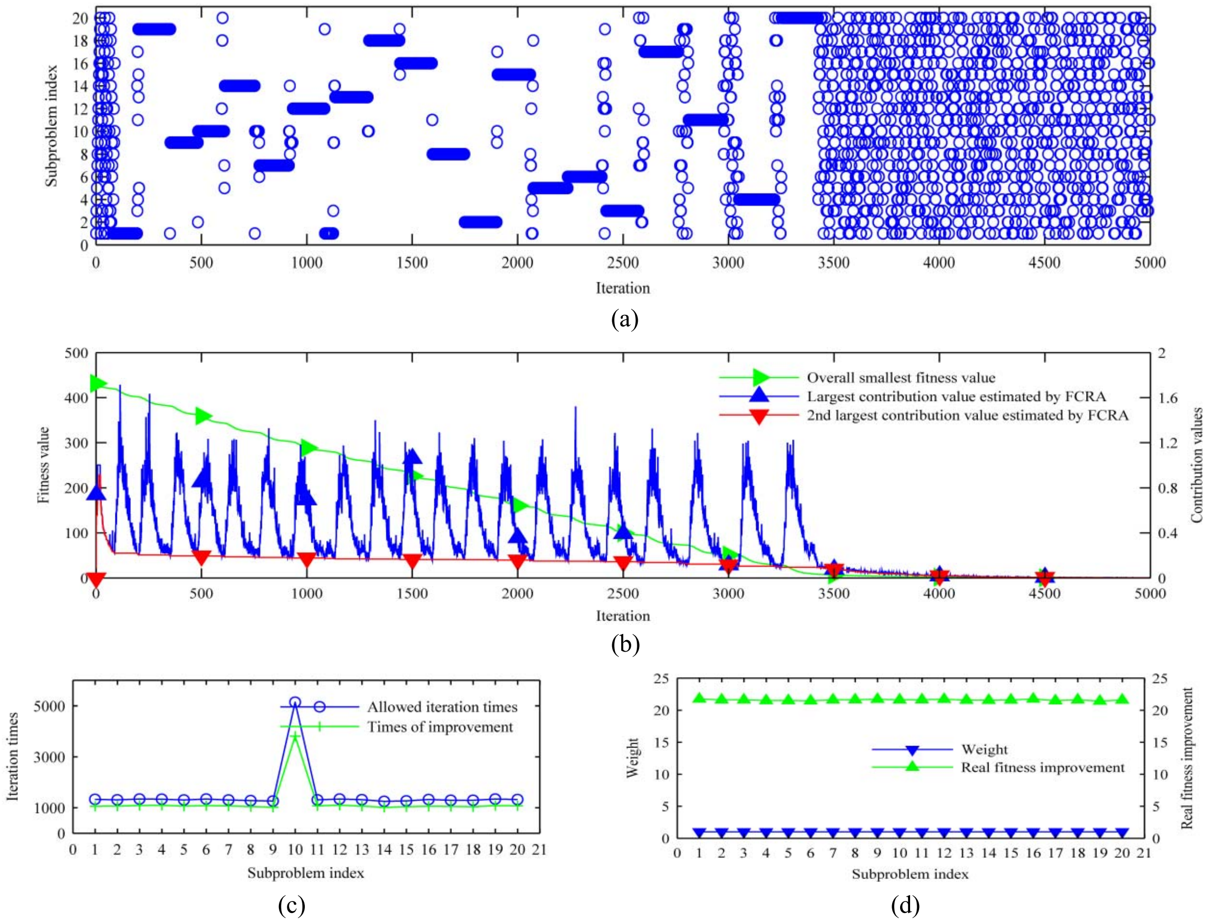

Fig. 2. Behavior of FCRACC on the CEC2010 function f_{16}. (a) Subproblem optimized at each of the first 5000 iterations. (b) Largest and the second largest contributions estimated at each iteration and the acquired smallest overall fitness value. (c) Allowed iteration times versus times of improvement. (d) Contribution weight versus real fitness improvement.

Fig. 3 demonstrates the CRA and optimization results of $\text{FCRACC}_{\text{ID+SHADE}}$ on the CEC2013 function f_8. This function can also be ideally decomposed into 20 subproblems, but these subproblems have different dimensions varying from 25 to 100 and different contribution weights of magnitude orders varying from 10^{-6} to 10^9 [50]. From Fig. 3(a), it can be observed that the third and the fifth subproblems, which have the largest and the second largest weights, respectively, undergo many more consecutive iterations in the first half and the second half of the whole search process, respectively, and the 11th and the 13th subproblems, which have the second smallest and the smallest weights, respectively, are only allocated with very few iterations in the second half of the search process.

From Fig. 3(b), we can see that the fitness value decreases rapidly in the first half of the search process, which mainly benefits from the fact that the significantly large weight of the third subproblem attracts many CRs for it. Besides, although the decreasing rate of the fitness value slows down in the second half of the search process, the largest contribution value estimated by FCRA still keeps a great order of magnitude, which means that the optimizer remains active and can still get better solutions.

From Fig. 3(c) and (d), the following observation can be made: the larger the weight of a subproblems, the more iterations it is allocated, and the larger contribution it makes to the total fitness improvement. One exception is that the fifth subproblem which has smaller weight than the third one is assigned with more iterations. The reason consists in that the third subproblem has much lower dimension. After the first half of the whole search process, the optimizer is near stagnant on this subproblem, while it is still active on the fifth one even at the end of the search process. This can be further verified by the fact that the optimizer achieves fitness improvement on the fifth subproblem at almost all iterations allocated to it.

The behavior of FCRACC on the two different kinds of functions indicates that FCRA can allocate CR to different subproblems adaptively and subtly according to their real-time contributions and evolution status, thus significantly increasing the utilization efficiency of CR. As a consequence, the performance of CC can be enhanced.

D. Comparison Between FCRACC and Other CC Algorithms Under Ideal Decomposition

To show the effectiveness of FCRA and SHADE, we compared two FCRACC algorithms ($\text{FCRACC}_{\text{ID+SaNSDE}}$ and $\text{FCRACC}_{\text{ID+SHADE}}$) with four other CC algorithms under

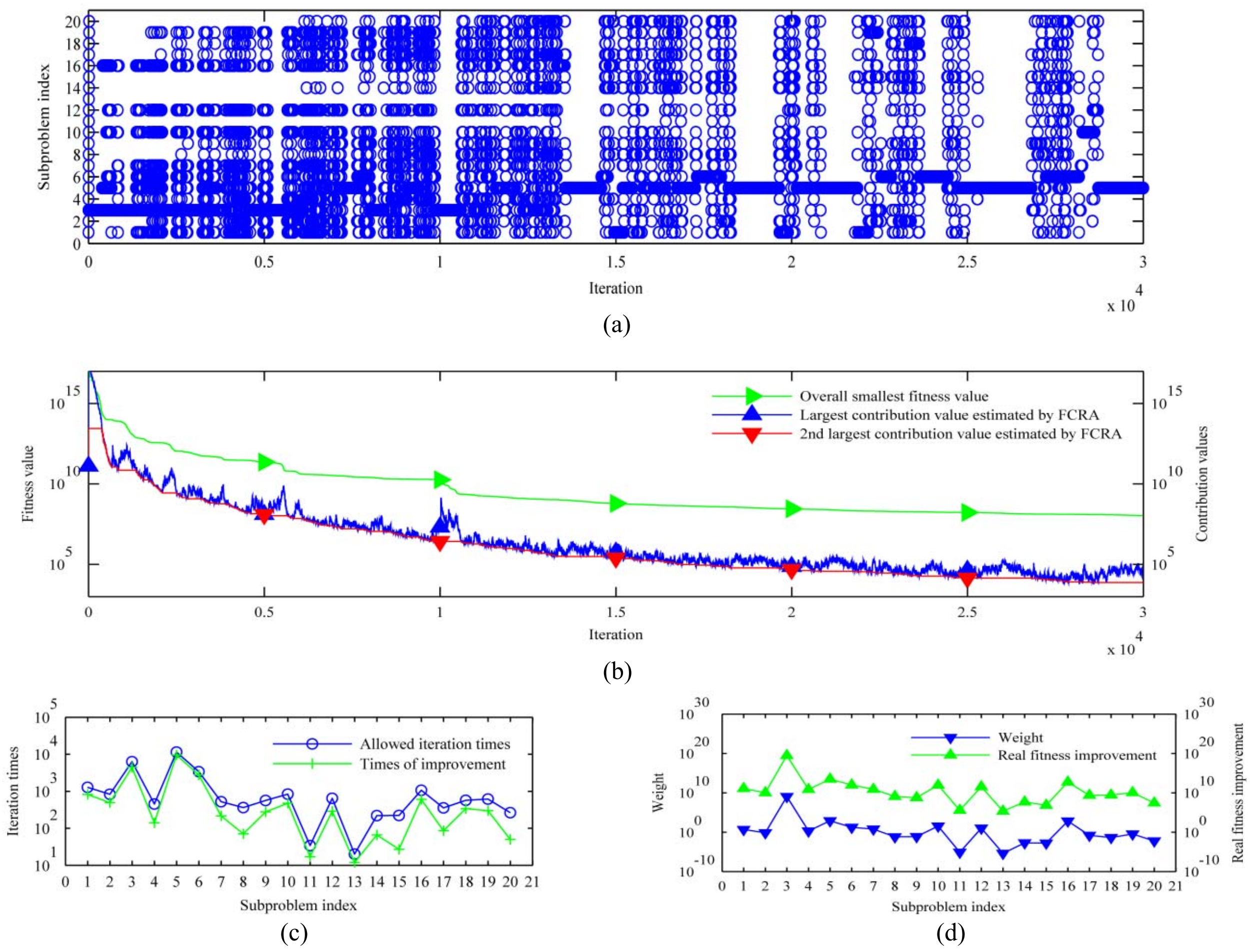

Fig. 3. Behavior of FCRACC on the CEC2013 function f_8. (a) Subproblem optimized at each iteration. (b) Largest and the second largest contributions estimated at each iteration and the acquired smallest overall fitness value. (c) Allowed iteration times versus times of improvement. (d) Contribution weight versus real fitness improvement.

ideal decomposition. These four algorithms include the traditional CC [7], CBCC1, CBCC2 [11], and the recently developed CCFR [15], all of which have been briefly introduced in Sections I and II. The difference between $\text{FCRACC}_{\text{ID+SaNSDE}}$ and the four existing CC algorithms mainly lies in the CRA strategy. Therefore, the comparison of their performance can highlight the advantage and disadvantage of FCRA. On the other hand, the optimizer is the only difference between $\text{FCRACC}_{\text{ID+SaNSDE}}$ and $\text{FCRACC}_{\text{ID+SHADE}}$, then their performance comparison can demonstrate the competitiveness of SHADE over SaNSDE in solving LSOPs within the CC framework.

Table II summarizes the average fitness values and the corresponding standard deviations obtained by the six CC algorithms on each function in the CEC2010 and CEC2013 benchmark suites. It is necessary to mention that to ensure the fairness of the comparison, the four existing CC algorithms were set with the same parameter values as given in their original papers, and their experimental results are directly taken from [15]. Table II also reports Cohen's d effect size [53] (within parentheses) to quantify the difference between the average fitness values of $\text{FCRACC}_{\text{ID+SaNSDE}}$ and those of the other five algorithms. Cohen's d effect size is independent of the sample size and is generally considered "small," "medium," and "large" if its absolute value belongs to [0.2, 0.3), [0.3, 0.8) and $[0.8, +\infty)$, respectively. Moreover, it is considered that there is no difference between two solutions if both of their fitness values are not greater than 1.0×10^{-10}. The superscripts "+," "∼," and "−" labeled in Table II denote that the performance of the corresponding algorithm is better than, similar to, and worse than that of $\text{FCRACC}_{\text{ID+SaNSDE}}$, respectively.

1) Comparison of the CEC2010 Suite: The results in Table II demonstrate that $\text{FCRACC}_{\text{ID+SaNSDE}}$ is rather successful. It outperforms the traditional CC, CBCC1, CBCC2, and CCFR on all three separable functions (f_1-f_3) except being surpassed by CCFR on f_2. It is worth mentioning that for these kinds of functions, the traditional CC and CCFR optimize all variables separately, CBCC optimizes them together [15], while FCRACC randomly divides them into groups of size 50 and optimizes the variable groups separately. For 15 partially separable functions (f_4–f_{18}), $\text{FCRACC}_{\text{ID+SaNSDE}}$ also shows exciting performance. It yields better solutions than all four existing algorithms on 11 functions and similar solutions on two functions. Although it shows performance deterioration on f_{10} and f_{15}, the average fitness values it obtains on them still have the same order of magnitude with the corresponding ones obtained by the competitors. The excellent performance

TABLE II
AVERAGE FITNESS VALUES ± STANDARD DEVIATIONS OBTAINED BY THE SIX CC ALGORITHMS ON THE CEC2010 AND CEC2013 FUNCTIONS OVER 25 INDEPENDENT RUNS, AND THE STATISTICAL RESULTS OBTAINED BASED ON COHEN'S *d* EFFECT SIZE AND FRIEDMAN TEST

	Fun.	$CC_{ID+SaNSDE}$	$CBCC1_{ID+SaNSDE}$	$CBCC2_{ID+SaNSDE}$	$CCFR_{ID+SaNSDE}$	$FCRACC_{ID+SaNSDE}$	$FCRACC_{ID+SHADE}$
CEC2010 functions	f_1	3.5e+11 ± 2.0e+10 (25.26) −	9.9e+06 ± 1.3e+07 (1.10) −	9.9e+06 ± 1.3e+07 (1.10) −	1.2e−05 ± 4.9e−06 (3.53) −	**6.94e−24 ± 1.46e−23**	**0.00e+00 ± 0.00e+00** (−) ≈
	f_2	9.4e+03 ± 2.1e+02 (63.84) −	4.7e+03 ± 4.8e+02 (13.89) −	4.7e+03 ± 4.8e+02 (13.89) −	2.7e+01 ± 5.2e+00 (−3.80) +	7.68e+01 ± 1.82e+01	**3.04e+00 ± 2.38e−01** (−5.85) +
	f_3	2.0e+01 ± 4.4e−02 (7.58) −	1.2e+01 ± 3.7e−01 (4.31) −	1.2e+01 ± 3.7e−01 (4.31) −	4.6e+00 ± 4.6e−01 (1.33) −	**1.30e+00 ± 3.56e+00**	**8.54e−01 ± 2.81e+00** (−0.14) ≈
	f_4	3.4e+14 ± 7.5e+13 (6.54) −	6.0e+10 ± 4.4e+10 (1.37) −	9.9e+10 ± 2.7e+10 (3.61) −	8.3e+10 ± 6.2e+10 (1.53) −	1.36e+10 ± 2.09e+10	**2.88e+07 ± 2.45e+07** (−0.94) +
	f_5	4.9e+08 ± 2.4e+07 (22.58) −	6.8e+07 ± 1.0e+07 (−0.07) ≈	6.7e+07 ± 9.1e+06 (−0.17) ≈	7.2e+07 ± 1.3e+07 (0.26) −	6.88e+07 ± 1.22e+07	**4.66e+07 ± 7.18e+06** (−2.26) +
	f_6	1.1e+07 ± 7.5e+05 (21.17) −	1.3e+06 ± 6.4e+05 (2.93) −	1.3e+06 ± 6.8e+05 (2.76) −	7.7e+05 ± 7.1e+05 (1.57) −	**7.87e−01 ± 2.78e+00**	**6.38e−01 ± 2.57e+00** (−0.06) ≈
	f_7	7.7e+10 ± 9.6e+09 (11.58) −	5.9e+04 ± 9.3e+03 (9.16) −	8.4e+04 ± 1.9e+04 (6.38) −	1.5e−03 ± 2.5e−04 (8.66) −	**1.11e−20 ± 8.72e−21**	**1.11e−21 ± 4.80e−22** (−) ≈
	f_8	1.8e+14 ± 9.3e+13 (2.79) −	8.6e+05 ± 1.6e+06 (0.40) −	1.0e+06 ± 1.7e+06 (0.49) −	**3.2e+05 ± 1.1e+06** (0.00) ≈	**3.19e+05 ± 1.10e+06**	**1.59e+05 ± 7.97e+05** (−0.17) ≈
	f_9	9.4e+08 ± 7.1e+07 (19.05) −	1.7e+07 ± 2.1e+07 (0.98) −	2.8e+09 ± 1.8e+09 (2.24) −	9.4e+06 ± 1.2e+06 (7.65) −	2.69e+06 ± 4.04e+05	**8.59e+05 ± 1.08e+05** (−6.32) +
	f_{10}	4.8e+03 ± 6.7e+01 (−6.70) +	3.0e+03 ± 1.7e+02 (−10.56) +	4.5e+03 ± 6.6e+02 (−5.07) +	**1.4e+03 ± 1.0e+02** (−14.52) +	7.67e+03 ± 6.15e+02	5.86e+03 ± 6.71e+02 (−2.87) +
	f_{11}	4.1e+01 ± 1.5e+00 (12.84) −	2.2e+01 ± 3.2e+00 (5.70) −	2.4e+01 ± 2.7e+00 (6.60) −	1.0e+01 ± 2.7e+00 (2.57) −	1.06e+00 ± 4.23e+00	**1.33e−05 ± 6.64e−05** (−0.36) +
	f_{12}	4.9e+05 ± 3.4e+04 (20.80) −	1.8e+04 ± 6.5e+03 (4.00) −	2.5e+04 ± 7.3e+03 (4.94) −	1.2e+00 ± 4.6e+00 (0.38) −	5.28e−06 ± 2.82e−06	**3.44e−10 ± 1.19e−10** (−2.70) +
	f_{13}	1.5e+07 ± 4.1e+06 (5.28) −	1.9e+04 ± 6.3e+03 (4.33) −	2.8e+04 ± 5.4e+03 (7.46) −	3.2e+02 ± 9.9e+01 (2.95) −	9.09e+01 ± 5.27e+01	**7.34e+00 ± 3.19e+00** (−2.28) +
	f_{14}	2.7e+07 ± 2.1e+06 (7.74) −	2.8e+07 ± 2.1e+06 (8.33) −	9.5e+09 ± 5.2e+08 (26.33) −	2.5e+07 ± 2.9e+06 (5.07) −	1.39e+07 ± 1.25e+06	**6.01e+06 ± 5.52e+05** (−8.33) +
	f_{15}	4.0e+03 ± 1.6e+02 (−10.37) +	4.0e+03 ± 1.5e+02 (−10.41) +	4.2e+03 ± 1.6e+02 (−9.91) +	**2.8e+03 ± 1.3e+02** (−13.26) +	8.53e+03 ± 6.10e+02	6.11e+03 ± 8.17e+02 (−3.43) +
	f_{16}	2.0e+01 ± 4.0e+00 (3.71) −	1.9e+01 ± 3.2e+00 (3.72) −	2.0e+01 ± 3.4e+00 (3.88) −	2.0e+01 ± 2.6e+00 (4.11) −	1.87e+00 ± 5.82e+00	**1.40e−13 ± 3.42e−15** (−0.46) +
	f_{17}	2.2e+01 ± 3.7e+01 (0.84) −	3.5e+01 ± 4.9e+01 (1.02) −	1.4e+02 ± 4.4e+01 (4.58) −	9.8e+00 ± 1.1e+01 (1.24) −	3.69e−01 ± 2.61e−01	**4.70e−03 ± 1.26e−03** (−2.01) +
	f_{18}	1.0e+03 ± 1.7e+02 (1.08) −	1.1e+03 ± 1.8e+02 (1.68) −	1.4e+03 ± 1.9e+02 (3.46) −	1.1e+03 ± 1.8e+02 (1.68) −	8.35e+02 ± 1.39e+02	**3.85e+02 ± 6.56e+01** (−4.23) +
	f_{19}	1.2e+06 ± 9.5e+04 (3.38) −	1.2e+06 ± 9.5e+04 (3.38) −	1.2e+06 ± 9.5e+04 (3.38) −	1.2e+06 ± 9.5e+04 (3.38) −	8.75e+05 ± 1.01e+05	**2.86e+05 ± 3.29e+04** (−8.00) +
	f_{20}	1.0e+09 ± 9.0e+08 (1.59) −	1.0e+09 ± 9.0e+08 (1.59) −	1.0e+09 ± 9.0e+08 (1.59) −	1.0e+09 ± 9.0e+08 (1.59) −	8.26e+06 ± 1.44e+07	**1.05e+06 ± 2.23e+06** (−0.71) +
	+/≈/− No.	2/0/18	2/1/17	2/1/17	3/1/16	−	15/5/0
CEC2013 functions	f_1	3.7e+11 ± 1.5e+10 (35.60) −	1.4e+07 ± 3.6e+07 (0.56) −	1.4e+07 ± 3.6e+07 (0.56) −	1.3e−05 ± 3.2e−06 (5.86) −	**1.45e−23 ± 4.62e−23**	**0.00e+00 ± 0.00e+00** (−) ≈
	f_2	8.5e+04 ± 5.1e+03 (24.02) −	2.1e+04 ± 9.9e+02 (30.42) −	2.1e+04 ± 9.9e+02 (30.42) −	**5.5e−01 ± 1.5e+00** (−6.85) +	1.26e+02 ± 2.64e+01	7.44e+00 ± 6.90e−01 (−6.48) +
	f_3	2.1e+01 ± 9.1e−03 (20.83) −	2.1e+01 ± 1.1e−02 (20.73) −	2.1e+01 ± 1.1e−02 (20.73) −	**2.0e+01 ± 3.1e−07** (−2.34) +	2.01e+01 ± 6.17e−02	2.01e+01 ± 3.93e−02 (0.00) ≈
	f_4	1.7e+12 ± 4.8e+11 (5.11) −	1.6e+08 ± 6.0e+07 (3.66) −	6.6e+10 ± 5.6e+09 (17.01) −	4.5e+07 ± 1.7e+07 (3.14) −	7.62e+06 ± 2.51e+06	**5.38e+05 ± 2.10e+05** (−4.06) +
	f_5	1.2e+07 ± 6.9e+05 (12.55) −	2.5e+06 ± 4.2e+05 (−4.62) +	**2.4e+06 ± 4.5e+05** (−4.70) +	2.5e+06 ± 2.7e+05 (−5.31) +	4.59e+06 ± 5.00e+05	3.47e+06 ± 2.77e+05 (−2.83) +
	f_6	1.1e+06 ± 1.6e+03 (30.06) −	1.1e+06 ± 1.9e+03 (27.65) −	1.1e+06 ± 1.7e+03 (29.23) −	1.1e+06 ± 1.2e+03 (33.49) −	**1.05e+06 ± 1.79e+03**	**1.05e+06 ± 5.70e+03** (0.00) ≈
	f_7	4.2e+09 ± 1.1e+09 (5.51) −	1.9e+07 ± 2.4e+07 (1.00) −	9.6e+07 ± 3.7e+08 (0.37) −	8.6e+06 ± 1.9e+07 (0.50) −	**1.54e+06 ± 7.67e+06**	**1.64e+06 ± 9.95e+05** (0.02) ≈
	f_8	4.7e+13 ± 2.8e+13 (2.42) −	2.0e+13 ± 2.8e+13 (1.03) −	1.0e+12 ± 1.3e+11 (11.08) −	9.6e+09 ± 1.6e+10 (0.66) −	2.25e+09 ± 1.61e+09	**1.50e+08 ± 3.98e+07** (−1.88) +
	f_9	2.9e+08 ± 5.2e+07 (−2.17) +	2.5e+08 ± 3.8e+07 (−3.40) +	2.2e+08 ± 2.8e+07 (−4.47) +	**1.9e+08 ± 2.8e+07** (−5.22) +	3.99e+08 ± 5.05e+07	3.02e+08 ± 2.39e+07 (−2.51) +
	f_{10}	9.4e+07 ± 2.9e+05 (4.80) −	9.4e+07 ± 2.8e+05 (4.89) −	9.4e+07 ± 2.3e+05 (5.38) −	9.5e+07 ± 1.9e+05 (10.26) −	9.27e+07 ± 2.62e+05	**9.22e+07 ± 3.82e+05** (−1.56) +
	f_{11}	2.2e+09 ± 8.4e+09 (0.29) −	3.0e+09 ± 1.0e+10 (0.36) −	4.9e+10 ± 9.5e+10 (0.74) −	**3.3e+08 ± 3.2e+08** (−0.39) +	4.83e+08 ± 4.59e+08	5.22e+08 ± 2.81e+08 (0.10) ≈
	f_{12}	6.1e+08 ± 7.1e+08 (1.22) −	6.1e+08 ± 7.1e+08 (1.22) −	6.1e+08 ± 7.1e+08 (1.22) −	6.0e+08 ± 7.1e+08 (1.20) −	1.08e+07 ± 3.84e+07	**6.30e+05 ± 2.11e+06** (−0.38) +
	f_{13}	9.5e+08 ± 5.4e+08 (1.58) −	9.5e+08 ± 5.4e+08 (1.58) −	9.5e+08 ± 5.4e+08 (1.58) −	9.3e+08 ± 5.3e+08 (1.56) −	3.41e+08 ± 1.28e+08	**1.41e+07 ± 1.44e+07** (−3.66) +
	f_{14}	2.2e+09 ± 2.1e+09 (1.31) −	2.2e+09 ± 2.1e+09 (1.31) −	2.2e+09 ± 2.1e+09 (1.31) −	2.1e+09 ± 2.1e+09 (1.24) −	2.70e+08 ± 3.12e+08	**4.82e+07 ± 8.67e+07** (−0.99) +
	f_{15}	8.3e+06 ± 3.3e+06 (1.49) −	8.3e+06 ± 3.3e+06 (1.49) −	8.3e+06 ± 3.3e+06 (1.49) −	8.2e+06 ± 3.3e+06 (1.44) −	4.68e+06 ± 1.22e+06	**1.11e+06 ± 2.08e+05** (−4.16) +
	+/≈/− No.	1/0/14	2/0/13	2/0/13	5/0/10	−	10/5/0
Ranking		5.1429	4.1429	4.5857	3.0143	2.5143	1.6000

† "+", "≈", and "−" denote that the performance of the corresponding algorithm is better than, similar to, and worse than that of $FCRACC_{ID+SaNSDE}$, respectively.
‡ The results marked in boldface indicate they are the best.

of $FCRACC_{ID+SaNSDE}$ stems from its ability to make better use of CR. It is notable that $FCRACC_{ID+SaNSDE}$ also performs better on f_{19} and f_{20} which are nonseparable functions. The reason is that the four existing CC algorithms re-evaluate the population of each subproblem at the beginning of each cycle, while FCRACC avoids this by evaluating the solution of a subproblem according to its fitness improvement to the best overall solution instead of its fitness itself (see Section III-C).

As for $FCRACC_{ID+SHADE}$, it performs no worse than $FCRACC_{ID+SaNSDE}$ on all CEC2010 functions. Especially for f_1, f_7, f_{12}, and f_{16}, it gets a near optimal solution in each of the 25 independent runs. This indicates that FCRACC is compatible with different optimizers and SHADE is more efficient than SaNSDE in solving LSOPs within CC.

2) Comparison on the CEC2013 Suite: Compared with the CEC2010 functions, the CEC2013 functions are much more difficult to solve, since the dimensions and the contribution weights of different subproblems are very imbalanced, and more complicated transformations are introduced into the base functions [50]. On the whole, most solutions reported in Table II are far away from corresponding optimal ones. Even so, the performance of the two FCRACC algorithms is satisfactory, compared with that of the four existing CC algorithms. $FCRACC_{ID+SaNSDE}$ outperforms the traditional CC, CBCC1, CBCC2, and CCFR on 14, 13, 13, and 10 functions, respectively. For f_2 and f_{11}, it is defeated by CCFR, but still provides significantly better solutions than the other three existing algorithms. When comparing $FCRACC_{ID+SaNSDE}$ with $FCRACC_{ID+SHADE}$, we can see that the latter improves the former on ten functions and shows similar performance on the remaining five functions. This further verifies the compatibility of FCRACC and the efficiency of SHADE.

The last row of Table II lists the ranking of the six CC algorithms on the CEC2010 and CEC2013 functions according to the Friedman test, from which it can be concluded that $FCRACC_{ID+SHADE}$ performs best, followed by $FCRACC_{ID+SaNSDE}$, CCFR, CBCC1, and CBCC2, whereas the traditional CC is definitely defeated by the other five algorithms.

E. Comparisons Among FCRACC, Other CC Algorithms, and Non-CC Algorithms

To evaluate the performance of FCRACC more comprehensively, we further tested the two FCRACC algorithms coupled with DG2 [35], which is an efficient decomposition method developed recently. Their experimental results are

TABLE III
AVERAGE FITNESS VALUES ± STANDARD DEVIATIONS OBTAINED BY THE SIX CC ALGORITHMS AND THE TWO NON-CC ALGORITHMS ON THE CEC2010 AND CEC2013 FUNCTIONS OVER 25 INDEPENDENT RUNS, AND THE STATISTICAL RESULTS OBTAINED BASED ON COHEN'S d EFFECT SIZE AND FRIEDMAN TEST

	Fun.	DECC-G$_{SaNSDE}$	DECC-D$_{SaNSDE}$	DECC$_{DG2+SaNSDE}$	CCFR$_{DG2+SaNSDE}$	MA-SW-Chains	MOS-CEC2013	FCRACC$_{DG2+SaNSDE}$	FCRACC$_{DG2+SHADE}$
CEC2010 functions	f_1	4.0e−07 ± 1.0e−07 $^{-}$	**1.0e−22 ± 9.0e−21** $^{\approx}$	1.7e+07 ± 2.1e+07 $^{-}$	2.0e−05 ± 7.0e−06 $^{-}$	**3.88e−14 ± 3.59e−14** $^{\approx}$	**0.00e+00 ± 0.00e+00** $^{\approx}$	**1.22e−23 ± 2.88e−23**	**0.00e+00 ± 0.00e+00** $^{\approx}$
	f_2	1.3e+03 ± 3.0e+01 $^{-}$	6.5e+01 ± 4.0e+01 $^{+}$	4.7e+03 ± 4.8e+02 $^{-}$	1.7e+02 ± 9.0e+00 $^{-}$	8.63e+02 ± 5.84e+01 $^{-}$	**0.00e+00 ± 0.00e+00** $^{+}$	1.12e+02 ± 2.00e+02	8.59e+01 ± 5.50e+00 $^{\approx}$
	f_3	1.1e+00 ± 4.0e−01 $^{+}$	2.3e+00 ± 2.0e−01 $^{+}$	1.2e+01 ± 3.7e−01 $^{+}$	1.2e+01 ± 4.0e−01 $^{+}$	**5.41e−13 ± 2.13e−13** $^{+}$	**1.65e−12 ± 6.73e−14** $^{+}$	1.30e+01 ± 1.03e+00	1.29e+01 ± 7.41e−01 $^{\approx}$
	f_4	2.0e+13 ± 5.0e+12 $^{-}$	3.0e+12 ± 9.0e+11 $^{-}$	8.9e+10 ± 4.6e+10 $^{-}$	1.0e+11 ± 8.0e+10 $^{-}$	2.94e+11 ± 9.32e+10 $^{-}$	1.56e+10 ± 6.02e+09 $^{\approx}$	1.43e+10 ± 1.85e+10	**1.08e+08 ± 1.33e+08** $^{+}$
	f_5	2.5e+08 ± 7.0e+07 $^{-}$	2.9e+08 ± 1.0e+08 $^{-}$	6.7e+07 ± 1.0e+07 $^{+}$	9.2e+07 ± 2.0e+07 $^{-}$	1.75e+08 ± 1.03e+08 $^{-}$	1.11e+08 ± 2.25e+07 $^{-}$	7.10e+07 ± 9.09e+06	**4.85e+07 ± 6.81e+06** $^{+}$
	f_6	5.3e+06 ± 1.0e+06 $^{-}$	5.9e+06 ± 5.0e+06 $^{-}$	6.4e+05 ± 6.8e+05 $^{-}$	6.8e+05 ± 7.0e+05 $^{-}$	3.52e+04 ± 1.72e+05 $^{-}$	**1.22e−07 ± 6.43e−08** $^{+}$	1.31e+01 ± 2.51e+00	1.29e+01 ± 1.73e+00 $^{\approx}$
	f_7	8.1e+08 ± 5.0e+08 $^{-}$	1.5e+05 ± 2.0e+05 $^{-}$	4.2e+04 ± 1.2e+04 $^{-}$	2.0e−03 ± 3.0e−04 $^{-}$	3.30e+02 ± 1.40e+03 $^{-}$	**0.00e+00 ± 0.00e+00** $^{\approx}$	**4.92e−20 ± 1.79e−20**	**8.92e−22 ± 4.06e−22** $^{\approx}$
	f_8	6.8e+07 ± 3.0e+07 $^{-}$	1.3e+08 ± 1.0e+08 $^{-}$	5.2e+05 ± 1.3e+06 $^{\approx}$	3.2e+05 ± 1.0e+06 $^{\approx}$	9.28e+06 ± 2.36e+07 $^{-}$	**1.95e+00 ± 8.03e+00** $^{+}$	3.32e+05 ± 1.10e+06	3.18e+05 ± 1.12e+06 $^{\approx}$
	f_9	4.5e+08 ± 5.0e+07 $^{-}$	1.0e+08 ± 9.0e+06 $^{-}$	5.4e+07 ± 6.4e+07 $^{-}$	1.3e+07 ± 2.0e+06 $^{-}$	1.45e+07 ± 1.59e+06 $^{-}$	3.46e+06 ± 4.49e+05 $^{+}$	3.66e+06 ± 5.91e+05	**1.26e+06 ± 1.35e+05** $^{+}$
	f_{10}	1.1e+04 ± 4.0e+02 $^{-}$	4.1e+03 ± 1.0e+03 $^{+}$	4.3e+03 ± 1.8e+02 $^{+}$	**1.8e+03 ± 1.0e+02** $^{+}$	2.06e+03 ± 1.19e+02 $^{+}$	3.78e+03 ± 1.47e+02 $^{+}$	8.05e+03 ± 6.22e+02	6.33e+03 ± 7.74e+02 $^{+}$
	f_{11}	2.6e+01 ± 1.0e+00 $^{-}$	1.0e+02 ± 1.0e+02 $^{-}$	2.3e+01 ± 2.1e+00 $^{-}$	2.0e+01 ± 3.0e+00 $^{-}$	3.69e+01 ± 8.24e+00 $^{-}$	1.91e+02 ± 4.07e−01 $^{-}$	1.30e+01 ± 1.05e+00	**8.74e+00 ± 6.50e−01** $^{+}$
	f_{12}	9.9e+04 ± 1.0e+04 $^{-}$	9.1e+03 ± 1.0e+03 $^{-}$	2.3e+04 ± 8.8e+03 $^{-}$	2.0e+01 ± 2.0e+01 $^{-}$	3.19e−06 ± 5.32e−07 $^{+}$	**0.00e+00 ± 0.00e+00** $^{+}$	3.50e−04 ± 1.53e−04	2.40e−07 ± 1.04e−07 $^{+}$
	f_{13}	5.3e+03 ± 3.0e+03 $^{-}$	5.4e+03 ± 3.0e+03 $^{-}$	2.5e+04 ± 7.8e+03 $^{-}$	5.3e+02 ± 1.0e+02 $^{-}$	1.09e+03 ± 6.29e+02 $^{-}$	7.14e+02 ± 5.68e+02 $^{-}$	1.99e+02 ± 6.58e+01	**8.31e+00 ± 1.16e+01** $^{+}$
	f_{14}	9.8e+08 ± 8.0e+07 $^{-}$	3.0e+08 ± 2.0e+07 $^{-}$	3.3e+07 ± 2.7e+06 $^{-}$	3.1e+07 ± 3.0e+06 $^{-}$	3.34e+07 ± 3.37e+06 $^{-}$	9.80e+06 ± 6.03e+05 $^{+}$	1.82e+07 ± 1.58e+06	**8.90e+06 ± 6.54e+05** $^{+}$
	f_{15}	1.2e+04 ± 7.0e+02 $^{-}$	1.3e+04 ± 2.0e+02 $^{-}$	4.4e+03 ± 1.9e+02 $^{+}$	3.2e+03 ± 2.0e+02 $^{+}$	**2.69e+03 ± 9.75e+01** $^{+}$	7.44e+03 ± 1.84e+02 $^{+}$	8.57e+03 ± 5.77e+02	6.44e+03 ± 6.95e+02 $^{+}$
	f_{16}	6.9e+01 ± 5.0e+00 $^{-}$	2.0e+02 ± 2.0e+02 $^{-}$	2.0e+01 ± 4.0e+00 $^{-}$	2.0e+01 ± 3.0e+00 $^{-}$	1.08e+02 ± 1.51e+01 $^{-}$	3.82e+02 ± 1.55e+01 $^{-}$	2.86e+00 ± 7.07e+00	**8.47e−01 ± 4.23e+00** $^{+}$
	f_{17}	3.1e+05 ± 2.0e+04 $^{-}$	7.5e+04 ± 5.0e+03 $^{-}$	8.0e+01 ± 5.2e+01 $^{-}$	6.7e+01 ± 9.0e+01 $^{-}$	1.26e+00 ± 9.45e−02 $^{+}$	**2.83e−07 ± 7.97e−08** $^{+}$	5.10e+00 ± 5.00e+00	2.35e−01 ± 5.98e−02 $^{+}$
	f_{18}	3.5e+04 ± 1.0e+04 $^{-}$	1.4e+04 ± 1.0e+04 $^{-}$	1.2e+03 ± 1.5e+02 $^{-}$	1.4e+03 ± 2.0e+02 $^{-}$	1.87e+03 ± 5.79e+02 $^{-}$	1.54e+03 ± 7.46e+02 $^{-}$	1.05e+03 ± 1.49e+02	**5.28e+02 ± 5.84e+01** $^{+}$
	f_{19}	1.1e+06 ± 6.0e+04 $^{-}$	1.6e+06 ± 1.0e+06 $^{-}$	1.3e+06 ± 1.0e+05 $^{-}$	1.3e+06 ± 1.0e+05 $^{-}$	2.85e+05 ± 1.74e+04 $^{+}$	**2.91e+04 ± 2.14e+03** $^{+}$	9.57e+05 ± 6.25e+04	3.46e+05 ± 3.67e+04 $^{+}$
	f_{20}	4.5e+03 ± 8.0e+02 $^{+}$	2.3e+03 ± 2.0e+02 $^{+}$	2.0e+09 ± 1.8e+09 $^{-}$	2.0e+09 ± 2.0e+09 $^{-}$	1.05e+03 ± 7.59e+01 $^{+}$	**3.52e+02 ± 4.43e+02** $^{+}$	2.17e+07 ± 3.28e+07	1.07e+06 ± 2.59e+06 $^{+}$
	+/≈/− No.	2/0/18	4/1/15	4/1/15	3/1/16	7/1/12	12/3/5	−	14/6/0
CEC2013 functions	f_1	3.0e−06 ± 2.0e−06 $^{-}$	**1.0e−17 ± 1.0e−17** $^{\approx}$	4.6e+07 ± 1.3e+08 $^{-}$	2.0e−05 ± 5.0e−06 $^{-}$	**8.49e−13 ± 1.09e−12** $^{\approx}$	**1.27e−22 ± 7.41e−23** $^{\approx}$	**5.26e−24 ± 1.20e−23**	**0.00e+00 ± 0.00e+00** $^{\approx}$
	f_2	1.3e+03 ± 3.0e+01 $^{-}$	**7.1e+01 ± 3.0e+01** $^{+}$	2.1e+04 ± 1.0e+03 $^{-}$	3.6e+02 ± 2.0e+01 $^{-}$	1.22e+03 ± 1.14e+02 $^{-}$	8.32e+02 ± 4.48e+01 $^{-}$	1.26e+02 ± 2.80e+01	1.35e+02 ± 5.07e+00 $^{-}$
	f_3	2.0e+01 ± 7.0e−03 $^{+}$	2.0e+01 ± 2.0e−03 $^{+}$	2.1e+01 ± 1.2e−02 $^{-}$	2.1e+01 ± 1.0e−02 $^{-}$	2.14e+01 ± 5.62e−02 $^{-}$	**9.18e−13 ± 5.12e−14** $^{+}$	2.07e+01 ± 8.18e−03	2.01e+01 ± 2.74e−03 $^{+}$
	f_4	2.0e+11 ± 1.0e+11 $^{-}$	3.0e+10 ± 2.0e+10 $^{-}$	2.9e+08 ± 9.7e+07 $^{-}$	9.6e+07 ± 4.0e+07 $^{-}$	4.58e+09 ± 2.46e+09 $^{-}$	1.74e+08 ± 7.87e+07 $^{-}$	1.24e+07 ± 5.00e+06	**1.04e+06 ± 3.11e+05** $^{+}$
	f_5	8.6e+06 ± 1.0e+06 $^{-}$	6.1e+06 ± 2.0e+06 $^{-}$	3.0e+06 ± 4.7e+05 $^{+}$	2.8e+06 ± 3.0e+05 $^{+}$	**1.87e+06 ± 3.06e+05** $^{+}$	6.94e+06 ± 8.85e+05 $^{-}$	4.68e+06 ± 5.10e+05	3.56e+06 ± 3.77e+05 $^{+}$
	f_6	1.1e+06 ± 1.0e+03 $^{-}$	1.1e+06 ± 2.0e+03 $^{-}$	1.1e+06 ± 1.6e+03 $^{-}$	1.1e+06 ± 1.0e+03 $^{-}$	1.01e+06 ± 1.53e+04 $^{+}$	**1.48e+05 ± 6.43e+04** $^{+}$	1.05e+06 ± 2.20e+03	1.05e+06 ± 2.48e+03 $^{\approx}$
	f_7	1.0e+09 ± 5.0e+08 $^{-}$	9.0e+07 ± 4.0e+07 $^{-}$	2.4e+07 ± 3.8e+07 $^{-}$	2.0e+07 ± 3.0e+07 $^{-}$	3.45e+06 ± 1.27e+06 $^{-}$	1.62e+04 ± 9.10e+03 $^{\approx}$	1.47e+04 ± 2.64e+04	**1.37e+03 ± 5.78e+03** $^{+}$
	f_8	9.0e+15 ± 4.0e+15 $^{-}$	2.0e+14 ± 9.0e+13 $^{-}$	7.4e+13 ± 5.8e+13 $^{-}$	7.0e+10 ± 1.0e+11 $^{-}$	4.85e+13 ± 1.02e+13 $^{-}$	8.00e+12 ± 3.07e+12 $^{-}$	5.05e+09 ± 4.90e+09	**2.22e+08 ± 7.92e+07** $^{+}$
	f_9	6.1e+08 ± 1.0e+08 $^{-}$	5.1e+08 ± 1.0e+08 $^{-}$	3.0e+08 ± 5.7e+07 $^{+}$	1.9e+08 ± 3.0e+07 $^{+}$	**1.07e+08 ± 1.68e+07** $^{+}$	3.83e+08 ± 6.29e+07 $^{+}$	3.99e+08 ± 3.43e+07	3.00e+08 ± 2.37e+07 $^{+}$
	f_{10}	9.3e+07 ± 5.0e+05 $^{-}$	9.3e+07 ± 6.0e+05 $^{-}$	9.5e+07 ± 3.0e+05 $^{-}$	9.5e+07 ± 2.0e+05 $^{-}$	9.18e+07 ± 1.06e+06 $^{+}$	**9.02e+05 ± 5.07e+05** $^{+}$	9.28e+07 ± 3.14e+05	9.22e+07 ± 3.03e+05 $^{+}$
	f_{11}	2.0e+11 ± 9.0e+10 $^{-}$	9.0e+08 ± 5.0e+08 $^{-}$	2.8e+09 ± 1.1e+10 $^{-}$	4.0e+08 ± 3.0e+08 $^{+}$	2.19e+08 ± 2.98e+07 $^{+}$	**5.22e+07 ± 2.05e+07** $^{+}$	5.74e+08 ± 4.91e+08	6.46e+08 ± 3.16e+08 $^{\approx}$
	f_{12}	4.4e+03 ± 7.0e+02 $^{+}$	2.3e+03 ± 2.0e+02 $^{+}$	1.6e+09 ± 1.6e+09 $^{-}$	1.6e+09 ± 2.0e+09 $^{-}$	1.25e+03 ± 1.05e+02 $^{+}$	**2.47e+02 ± 2.54e+02** $^{+}$	3.54e+07 ± 8.45e+07	8.12e+05 ± 1.15e+06 $^{+}$
	f_{13}	9.6e+09 ± 3.0e+09 $^{-}$	1.7e+09 ± 5.0e+08 $^{-}$	1.2e+09 ± 6.0e+08 $^{-}$	1.2e+09 ± 6.0e+08 $^{-}$	1.98e+07 ± 1.82e+06 $^{+}$	**3.40e+06 ± 1.06e+06** $^{+}$	5.75e+08 ± 4.01e+08	1.51e+07 ± 7.92e+06 $^{+}$
	f_{14}	2.0e+11 ± 5.0e+10 $^{-}$	7.4e+09 ± 9.0e+09 $^{-}$	3.5e+09 ± 3.2e+09 $^{-}$	3.4e+09 ± 3.0e+09 $^{-}$	1.36e+08 ± 2.11e+07 $^{+}$	**2.56e+07 ± 7.94e+06** $^{+}$	7.12e+08 ± 6.68e+08	4.87e+07 ± 5.85e+07 $^{+}$
	f_{15}	1.2e+07 ± 1.0e+06 $^{-}$	6.9e+06 ± 7.0e+05 $^{-}$	9.9e+06 ± 3.7e+06 $^{-}$	9.8e+06 ± 4.0e+06 $^{-}$	5.71e+06 ± 7.57e+05 $^{-}$	2.35e+06 ± 1.94e+05 $^{+}$	5.04e+06 ± 8.68e+05	**1.39e+06 ± 2.37e+05** $^{+}$
	+/≈/− No.	2/0/13	3/1/11	2/0/13	3/0/12	8/1/6	9/2/4	−	11/3/1
Ranking		6.7571	5.9286	5.7429	4.6143	4.0000	2.7571	3.7000	2.5000

† "+", "≈", and "−" denote that the performance of the corresponding algorithm is better than, similar to, and worse than that of FCRACC$_{DG2+SaNSDE}$, respectively.
‡ The results marked in boldface indicate they are the best.

presented in Table III, where the results of four well-known CC algorithms and two state-of-the-art non-CC algorithms are also reported for comparison. The four existing CC algorithms, that is, DECC-G [19], DECC-D [30], DECC [31], and CCFR [15], have been briefly introduced in Section II, where the first two employ RG and delta grouping as a decomposition method, respectively. For DECC and CCFR, their versions taking DG2 as decomposition method were employed in this experiment. The two non-CC algorithms, that is, MA-SW-Chains [51] and MOS-CEC2013 [52], are both memetic algorithms. They were ranked first in the IEEE CEC2010 and CEC2013 competitions on LSOP, respectively, [54]. It is necessary to emphasize that the source code of DG2 shared by Omidvar [55] was used in this experiment, the FEs consumed during the decomposition process were counted into the maximum-allowed FE number, and the experimental results of the six existing algorithms are taken from [15] and its supplementary materials, where these algorithms were set with the parameter values suggested by their original papers.

From Table III, it can be observed that FCRACC$_{\mathrm{DG2+SaNSDE}}$ outperforms DECC-G and DECC-D on 31 and 26 out of a total of 35 functions, respectively. The excellent result benefits from two factors: 1) DG2 provides more accurate decomposition results than RG and delta grouping and 2) FCRA allocates CR more economically and reasonably. Most of the functions on which FCRACC$_{\mathrm{DG2+SaNSDE}}$ does not show significant advantage are separable or nonseparable ones. For both kinds of functions, DG2 takes no effect and the effectiveness of FCRA is also weakened. Table III also shows that FCRACC$_{\mathrm{DG2+SaNSDE}}$ performs better than DECC and CCFR by several orders of magnitude on most test functions in terms of the average fitness value. For functions on which FCRACC$_{\mathrm{DG2+SaNSDE}}$ is defeated, its results are still in the same orders of magnitude with the corresponding ones of the other two algorithms. This again verifies the superiority of FCRA since the difference of these three algorithms only lies in CRA.

When comparing FCRACC$_{\mathrm{DG2+SaNSDE}}$ with the two memetic algorithms, we can see that FCRACC$_{\mathrm{DG2+SaNSDE}}$ has an edge over MA-SW-Chains since it performs no worse than the latter on 13 out of 20 CEC2010 functions and 7 out of 15 CEC2013 functions, but is outperformed by MOS-CEC2013 since the latter obtains better solutions on more functions. A more detailed investigation demonstrates that FCRACC$_{\mathrm{DG2+SaNSDE}}$ is defeated by both two memetic algorithms on all of the six nonseparable functions except

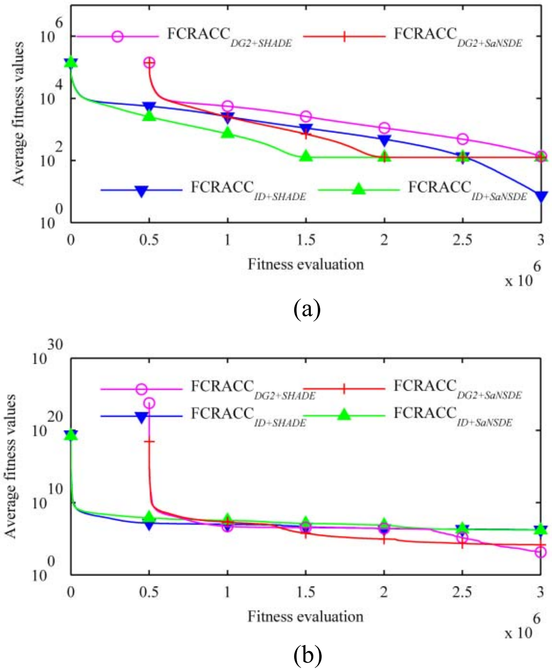

Fig. 4. Evolution curves of the four FCRACC algorithms. (a) CEC2013 function f_2. (b) CEC2013 function f_7.

that it shows better performance than MA-SW-Chains on the CEC2013 function f_{15}. The failure can be attributed to two reasons: 1) DG2 wastes part of CR on nonseparable functions and 2) the optimizer of $\text{FCRACC}_{\text{DG2+SaNSDE}}$, that is, SaNSDE, is inferior to the two memetic algorithms in solving LSOPs.

As for $\text{FCRACC}_{\text{DG2+SHADE}}$, the comparison result between it and $\text{FCRACC}_{\text{DG2+SaNSDE}}$ is similar to the one between $\text{FCRACC}_{\text{ID+SHADE}}$ and $\text{FCRACC}_{\text{ID+SaNSDE}}$, that is, it shows better performance than $\text{FCRACC}_{\text{DG2+SaNSDE}}$ on most functions. The main difference lies in two functions: the CEC2013 functions f_2 and f_7. For f_2, $\text{FCRACC}_{\text{DG2+SHADE}}$ performs worse than $\text{FCRACC}_{\text{DG2+SaNSDE}}$, although $\text{FCRACC}_{\text{ID+SHADE}}$ significantly outperforms $\text{FCRACC}_{\text{ID+SaNSDE}}$. This can be illustrated by Fig. 4(a), which shows that both $\text{FCRACC}_{\text{ID+SaNSDE}}$ and $\text{FCRACC}_{\text{DG2+SaNSDE}}$ stagnate before exhausting available CR, while $\text{FCRACC}_{\text{DG2+SHADE}}$ is precluded from finding better solutions by the fewer CR left for optimization. As for f_7, Fig. 4(b) shows the evolution curves of the four algorithms. They can be ranked as follows: $\text{FCRACC}_{\text{DG2+SHADE}}$, $\text{FCRACC}_{\text{DG2+SaNSDE}}$, $\text{FCRACC}_{\text{ID+SaNSDE}}$, and $\text{FCRACC}_{\text{ID+SHADE}}$. This result is somewhat counterintuitive since the ranking between $\text{FCRACC}_{\text{SHADE}}$ and $\text{FCRACC}_{\text{SaNSDE}}$ changes when different decomposition methods are used and the two algorithms with fewer CR for optimization outperform their respective counterparts. The main reason consists in that the fourth nonseparable subproblem obtained by ideal decomposition has the smallest contribution weight but the highest dimension [50]. It consumes most CR, which makes some other subproblems not fully optimized. This issue is more serious in $\text{FCRACC}_{\text{ID+SHADE}}$ as SHADE adopts a larger population size than SaNSDE, but is alleviated in $\text{FCRACC}_{\text{DG2+SHADE}}$ and $\text{FCRACC}_{\text{DG2+SaNSDE}}$

TABLE IV
STATISTICAL RESULTS OF THE FOUR ALGORITHMS

Fun.	Indicator	MA-SW-Chains	MOS-CEC2013	$\text{FCRACC}_{DG2+SaNSDE}$	$\text{FCRACC}_{DG2+SHADE}$
All the 35 functions	+/≈/− No.	11/2/22	16/3/16	1/9/25	−
	Ranking	3.0571	2.0429	3.0143	1.8857
29 (partially) separable functions	+/≈/− No.	8/2/19	11/3/15	1/9/19	−
	Ranking	3.1379	2.2241	2.8448	1.7931

since DG2 combines the fourth nonseparable subproblem with the first one which has the third largest contribution weight.

Table IV further summarizes the experimental results of $\text{FCRACC}_{\text{DG2+SaNSDE}}$, $\text{FCRACC}_{\text{DG2+SHADE}}$, and the two memetic algorithms. It can be seen that $\text{FCRACC}_{\text{DG2+SHADE}}$ not only outperforms $\text{FCRACC}_{\text{DG2+SaNSDE}}$, but also surpasses MA-SW-Chains and shows comparable performance to MOS-CEC2013 on most test functions. When the six nonseparable functions are excluded, the advantages of $\text{FCRACC}_{\text{DG2+SHADE}}$ are further highlighted. It performs no worse than MA-SW-Chains and MOS-CEC2013 on 21 and 18 out of total 29 (partially) separable functions, respectively. The rankings obtained through Friedman tests also indicate that $\text{FCRACC}_{\text{DG2+SHADE}}$ performs best among all of the algorithms tested.

V. CONCLUSION

In this paper, a novel CC framework called FCRACC is proposed for LSOPs. FCRACC is characterized by employing an FCRA strategy. Different from traditional CC, which equally allocates CR among subproblems and existing CBCCs which carry out CRA according to some heuristic rules, FCRACC allocates CR based on the theoretically optimal solution of an explicit CRA model. Moreover, it also fully considers the evolution characteristics of CC. Benefiting from these features, FCRACC can subtly allocate each iteration to the subproblem which is most likely to make the largest contribution to the total fitness improvement at a new iteration. Experimental results on two benchmark suites demonstrate that FCRACC is compatible with different optimizers, robust to the change of its parameter, and can achieve desirable CRA results and significantly outperform CC algorithms employing the same optimizer. When coupled with SHADE, FCRACC demonstrates highly competitive performance, even compared with state-of-the-art non-CC algorithms.

So far, most CC frameworks employ the same optimizer with the same population size for all subproblems which may have strikingly different characteristics. Our future work will focus on developing CC frameworks, which can adaptively configure optimizers, and studying CRA strategies for them.

REFERENCES

[1] Z.-H. Zhou, N. V. Chawla, Y. Jin, and G. J. Williams, “Big data opportunities and challenges: Discussions from data analytics perspectives,” *IEEE Comput. Intell. Mag.*, vol. 9, no. 4, pp. 62–74, Nov. 2014.

[2] S. Cheng, B. Liu, Y. Shi, Y. Jin, and B. Li, “Evolutionary computation and big data: Key challenges and future directions,” in *Proc. Int. Conf. Data Min. Big Data*, 2016, pp. 3–14.

[3] N. R. Sabar, J. Abawajy, and J. Yearwood, “Heterogeneous cooperative co-evolution memetic differential evolution algorithm for big data optimization problems,” *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 315–327, Apr. 2017.

[4] R. E. Bellman, *Dynamic Programming* (Dover Books on Mathematics). Santa Monica, CA, USA: Princeton Univ. Press, 1957.

[5] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution," in *Proc. IEEE Congr. Evol. Comput.*, 2001, pp. 1101–1108.

[6] F. van den 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.

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

[8] M. A. Potter and K. A. De Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evol. Comput.*, vol. 8, no. 1, pp. 1–29, Mar. 2000.

[9] S. Mahdavi, M. E. Shiri, and S. Rahnamayan, "Metaheuristics in large-scale global continues optimization: A survey," *Inf. Sci.*, vol. 295, pp. 407–428, Feb. 2015.

[10] G. A. Trunfio, "Metaheuristics for continuous optimization of high-dimensional problems: State of the art and perspectives," in *Big Data Optimization: Recent Developments and Challenges*, A. Emrouznejad, Ed. Cham, Switzerland: Springer, 2016, pp. 437–460.

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

[12] B. Kazimipour, M. N. Omidvar, X. Li, and A. K. Qin, "A sensitivity analysis of contribution-based cooperative co-evolutionary algorithms," in *Proc. IEEE Congr. Evol. Comput.*, May 2015, pp. 417–424.

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

[14] G. A. Trunfio, "Adaptation in cooperative coevolutionary optimization," in *Adaptation and Hybridization in Computational Intelligence*, I. Fister and I. Fister, Jr., Ed. Cham, Switzerland: Springer, 2015, pp. 91–109.

[15] 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.

[16] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 71–78.

[17] F. van den Bergh and A. P. Engelbrecht, "Cooperative learning in neural networks using particle swarm optimizers," *South Afr. Comput. J.*, vol. 26, pp. 84–90, Nov. 2000.

[18] Y. Shi, H.-F. Teng, and Z. Li, "Cooperative co-evolutionary differential evolution for function optimization," in *Proc. Int. Conf. Nat. Comput.*, 2005, pp. 1080–1088.

[19] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Inf. Sci.*, vol. 178, no. 15, pp. 2986–2999, Aug. 2008.

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

[21] M. N. Omidvar, X. Li, Z. Yang, and X. Yao, "Cooperative co-evolution for large scale optimization through more frequent random grouping," in *Proc. IEEE Congr. Evol. Comput.*, 2010, pp. 1754–1761.

[22] K. Weicker and N. Weicker, "On the improvement of coevolutionary optimizers by learning variable interdependencies," in *Proc. IEEE Congr. Evol. Comput.*, 1999, pp. 1627–1632.

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

[24] L. Sun, S. Yoshida, X. Cheng, and Y. Liang, "A cooperative particle swarm optimizer with statistical variable interdependence learning," *Inf. Sci.*, vol. 186, no. 1, pp. 20–39, Mar. 2012.

[25] H. Ge, L. Sun, X. Yang, S. Yoshida, and Y. Liang, "Cooperative differential evolution with fast variable interdependence learning and cross-cluster mutation," *Appl. Soft Comput.*, vol. 36, pp. 300–314, Nov. 2015.

[26] W. Chen and K. Tang, "Impact of problem decomposition on cooperative coevolution," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, 2013, pp. 733–740.

[27] H. Ge, L. Sun, G. 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.

[28] T. Ray and X. Yao, "A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning," in *Proc. IEEE Congr. Evol. Comput.*, May 2009, pp. 983–989.

[29] S.-F. Lin and Y.-C. Cheng, "A separability detection approach to cooperative particle swarm optimization," in *Proc. 7th Int. Conf. Nat. Comput.*, 2011, pp. 1141–1145.

[30] M. N. Omidvar, X. Li, and X. Yao, "Cooperative co-evolution with delta grouping for large scale non-separable function optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2010, pp. 1762–1769.

[31] M. N. Omidvar, X. 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.

[32] 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, p. 13, Jun. 2016.

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

[34] X.-M. Hu, F.-L. He, W.-N. Chen, and J. Zhang, "Cooperation coevolution with fast interdependency identification for large scale optimization," *Inf. Sci.*, vol. 381, pp. 142–160, Mar. 2017.

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

[36] Y. Sun, M. Kirley, and S. K. Halgamuge, "Quantifying variable interactions in continuous optimization problems," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 249–264, Apr. 2017.

[37] X. Li and X. Yao, "Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms," in *Proc. IEEE Congr. Evol. Comput.*, May 2009, pp. 18–21.

[38] X. 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.

[39] S. S. Aote, M. M. Raghuwanshi, and L. G. Malik, "A new particle swarm optimizer with cooperative coevolution for large scale optimization," in *Proc. 3rd Int. Conf. Front. Intell. Comput. Theory Appl.*, 2014, pp. 781–789.

[40] Y. Ren and Y. Wu, "An efficient algorithm for high-dimensional function optimization," *Soft Comput.*, vol. 17, no. 6, pp. 995–1004, Jun. 2013.

[41] X.-W. Zheng, D.-J. Lu, X.-G. Wang, and H. Liu, "A cooperative coevolutionary biogeography-based optimizer," *Appl. Intell.*, vol. 43, no. 1, pp. 95–111, Jul. 2015.

[42] J. Liu and K. Tang, "Scaling up covariance matrix adaptation evolution strategy using cooperative coevolution," in *Proc. Int. Conf. Intell. Data Eng. Autom. Learn.*, 2013, pp. 350–357.

[43] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, Jun. 2001.

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

[45] J. 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.

[46] L. Tang, Y. Dong, and J. Liu, "Differential evolution with an individual-dependent mechanism," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 560–574, Aug. 2015.

[47] X. Peng, K. Liu, and Y. Jin, "A dynamic optimization approach to the design of cooperative co-evolutionary algorithms," *Knowl. Based Syst.*, vol. 109, pp. 174–186, Oct. 2016.

[48] C. García-Martínez, P. D. Gutiérrez, D. Molina, M. Lozano, and F. Herrera, "Since CEC 2005 competition on real-parameter optimisation: A decade of research, progress and comparative analysis's weakness," *Soft Comput.*, vol. 21, no. 19, pp. 5573–5583, Oct. 2017.

[49] 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., 2010.

[50] 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," Evol. Comput. Mach. Learn. Group, RMIT Univ., Melbourne, VIC, Australia, Rep., 2013.

[51] D. Molina, M. Lozano, and F. Herrera, "MA-SW-Chains: Memetic algorithm based on local search chains for large scale continuous global optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2010, pp. 1–8.

[52] A. LaTorre, S. Muelas, and J.-M. Peña, “Large scale global optimization: Experimental results with MOS-based hybrid algorithms,” in *Proc. IEEE Congr. Evol. Comput.*, 2013, pp. 2742–2749.

[53] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*, 2nd ed. Hillsdale, NJ, USA: Lawrence Erlbaum, 1988.

[54] A. LaTorre, S. Muelas, and J.-M. Peña, “A comprehensive comparison of large scale global optimizers,” *Inf. Sci.*, vol. 316, pp. 517–549, Sep. 2015.

[55] M. N. Omidvar, *Homepage—Dr. Mohammad Nabi Omidvar*. Accessed: Jul. 18, 2017. [Online]. Available: http://nomidvar.com/

Zhigang Ren (M’12) received the B.Sc. degree in information and computing science from Shandong University, Jinan, China, in 2004, and the Ph.D. degree in control science and engineering from Xi’an Jiaotong University, Xi’an, China, in 2010.

Since 2011, he has been with the School of Electronic and Information Engineering, Xi’an Jiaotong University, where he is currently an Associate Professor. His current research interests include machine learning, intelligent computing, and modeling, optimization, and control of complex systems.

Yongsheng Liang (S’17) received the B.Eng. degree in automation from Xi’an Jiaotong University, Xi’an, China, in 2015, where he is currently pursuing the Ph.D. degree.

He is currently a Research Assistant with the School of Electronic and Information Engineering, Xi’an Jiaotong University. His current research interests include machine learning, intelligent computing, and large-scale optimization.

Aimin Zhang (M’12) received the B.Eng. degree in autocontrol, the M.Eng. degree in control science and engineering, and the Ph.D. degree in electrical engineering from Xi’an Jiaotong University, Xi’an, China, in 1983, 1989, and 2008, respectively.

Since 1983, she has been with the School of Electronic and Information Engineering, Xi’an Jiaotong University, where she is currently a Professor. Her current research interests include modeling, optimization, and control of complex systems, and nonlinear control theory and its applications.

Yang Yang received the B.Eng. degree in information engineering and the Ph.D. degree in control science and engineering from Xi’an Jiaotong University, Xi’an, China, in 2005 and 2011, respectively, and the Ph.D. degree in system innovation engineering from Tokushima University, Tokushima, Japan, in 2011.

She is currently an Associate Professor with the School of Electronic and Information Engineering, Xi’an Jiaotong University. Her current research interests include machine vision, pattern recognition, and big data processing.

Zuren Feng (M’05) received the M.Eng. and Ph.D. degrees in information and control engineering from Xi’an Jiaotong University, Xi’an, China, in 1982 and 1988, respectively.

Since 1994, he has been a Professor with the School of Electronic and Information Engineering, Xi’an Jiaotong University. In 1992, he was a Visiting Scholar with INRIA, Le Chesnay, France, researching manipulator control with flexible joints and the applications of Petri Nets in DEDS. In 1994, he was invited by the University of Kassel, Kassel, Germany, for research on mobile service robots. In 2006 and 2007, he was a Visiting Professor with the University of Bradford, Bradford, U.K. His current research interests include robotics and automation, machine vision and hearing, and intelligent computing.

Lin Wang (M’17) received the B.Eng. degree in information engineering and the Ph.D. degree in control science and engineering from Xi’an Jiaotong University, Xi’an, China, in 2004 and 2012, respectively.

He is currently an Associate Professor with the School of Information Science and Technology, Northwest University, Xi’an. His current research interests include point cloud processing and swarm intelligence optimization.