Efficient Resource Allocation in Cooperative Co-evolution for Large-scale Global Optimization

Ming Yang, Mohammad Nabi Omidvar, Changhe Li, *Member, IEEE*, Xiaodong Li, *Senior Member, IEEE*, Zhihua Cai, Borhan Kazimipour, and Xin Yao, *Fellow, IEEE*

***Abstract*—Cooperative Co-evolution (CC) is an explicit means of problem decomposition in multi-population evolutionary algorithms for solving large-scale optimization problems. For CC, subpopulations representing subcomponents of a large-scale optimization problem co-evolve, and are likely to have different contributions to the improvement of the best overall solution to the problem. Hence it makes sense that more computational resources should be allocated to the subpopulations with greater contributions. In this paper, we study how to allocate computational resources in this context and subsequently propose a new CC framework named CCFR to efficiently allocate computational resources among the subpopulations according to their dynamic contributions to the improvement of the objective value of the best overall solution. Our experimental results suggest that CCFR can make efficient use of computational resources and is a highly competitive CC framework for solving large-scale optimization problems.**

***Index Terms*—Cooperative co-evolution, resource allocation, problem decomposition, large-scale global optimization.**

I. Introduction

EVOLUTIONARY algorithms (EAs) have achieved a great success on solving many optimization problems [1]. However, they often lose their efficacy as the dimensionality of a problem increases [2]. Many real-world problems involve a large number of decision variables, e.g., the design of airfoil where thousands of variables are required to represent the complex shape of an aircraft wing [3]. This sort of large-scale optimization problems poses a serious challenge to existing EAs.

A natural approach to solving high-dimensional optimization problems is to employ the *divide-and-conquer* strategy [4]–[6], which decomposes a large-scale optimization problem into a set of smaller and simpler subproblems. These subproblems can be solved separately. The fully separable large-scale optimization problems, where there is no interdependence among decision variables, can be solved by optimizing each variable independently [7]. At the other end of the spectrum, the fully nonseparable large-scale optimization problems, where there is interdependence between any pair of variables, would need to be solved by optimizing all the variables together. However, most real-world problems fall somewhere between these two extremes, i.e., only some variables are independent or interdependent among each other [8]. For such partially separable problems, there are usually several clusters of interdependent variables. Cooperative Co-evolution (CC) [7] is an explicit means of problem decomposition in EAs. For CC, there is a set of subpopulations each of which is responsible for optimizing a subset of variables (i.e., a subcomponent).

Given a fixed computational budget, the performance of CC may be affected by how the computational resources are allocated among subpopulations [9]. For CC, different subpopulations are likely to make different amounts of contributions to the improvement of the best overall objective value (i.e., the objective value of the best overall solution consisting of the best individuals from these subpopulations). To be more computationally efficient, more computational resources should be allocated to the subpopulations that make greater contributions. It is shown in [9] that for imbalanced problems, where different subpopulations have unequal contributions to the overall objective value, a contribution-based cooperative co-evolution (CBCC) outperforms the traditional CC. However, for CBCC, the contribution information is accumulated from the beginning of the evolutionary process. CBCC relies much on the contribution information in the early stage of the evolutionary process, hence it may respond too slowly or even incorrectly to the local changes of the overall objective value. Since the contributions of subpopulations may change over time, it makes sense that the resource allocation should be done adaptively in real-time.

In this paper, we study how to allocate computational resources among subpopulations and propose a new CC framework, which can adaptively allocate computational resources to each subpopulation according to its dynamic contributions to the improvement of the best overall objective value. This new CC framework differs from existing CC frameworks in the following two aspects.

1) This new CC framework can check whether a subpop-

The work was supported in part by the National Natural Science Foundation of China (Grant Nos. 61305086, 61673355, 61673354, 61329302 and 61305079) and EPSRC (Grant No. EP/K001523/1). *(Corresponding author: Xin Yao.)*

M. Yang and Z. Cai are with the School of Computer Science, China University of Geosciences, Wuhan, 430074, China. M. Yang and Z. Cai are also with the Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan 430074, China (e-mail: yangming0702@gmail.com, zhcai@cug.edu.cn).

C. Li is with the School of Automation, China University of Geosciences, Wuhan, 430074, China (e-mail: changhe.lw@gmail.com).

M. N. Omidvar and X. Yao are with the Centre of Excellence for Research in Computational Intelligence and Applications, School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K. X. Yao is also with Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, 518055, China (e-mail: m.omidvar@cs.bham.ac.uk, x.yao@cs.bham.ac.uk).

X. Li and B. Kazimipour are with the School of Computer Science and Information Technology, RMIT University, Melbourne, VIC 3001, Australia (e-mail: xiaodong.li@rmit.edu.au, borhan.kazimipour@rmit.edu.au).

ulation is stagnant. To save computational resources, the stagnant subpopulations are excluded from evolution (see Sect. III-A).

2) In this new CC framework, the contribution of a subpopulation is updated dynamically. In each cycle, only the subpopulation with the greatest contribution is selected to undergo evolution (see Sect. III-B).

The remainder of this paper is organized as follows. Sect. II presents an overview of CC. Sect. III introduces our new CC framework. Sect. IV presents the experimental studies. Finally, Sect. V provides the concluding remarks.

II. Related Work

In the literature of evolutionary computation, the interdependence between decision variables of a problem is known as *linkage* [10] or *epistasis* [11]. The performance of a CC algorithm is greatly affected by the interdependence between variables [7], [12]. Variable grouping methods aiming to group interdependent variables into the same subcomponent being optimized play a key role in overcoming such a problem [13]. It is shown in [14] that if all the subcomponents are separable, the overall solution to the original problem is the combination of the respective solutions to all the subproblems. Here, we review CC mainly in the context of large-scale optimization.

In the original cooperatively co-evolutionary genetic algorithm (CCGA) proposed by Potter and De Jong [7], a D-dimensional problem is decomposed into D one-dimensional subproblems. CCGA then solves the subproblems using an evolutionary optimizer in a round-robin fashion. The experimental results in [7] show that the original CC cannot perform well on nonseparable functions, i.e., functions with interdependent variables, such as *Griewank* and *Rosenbrock*. Liu et al. [2] applied CC to fast evolutionary programming to solve large-scale optimization problems with up to 1000 dimensions. Van den Bergh and Engelbrecht [15] applied CC to particle swarm optimization (PSO) [16] and proposed a cooperatively co-evolutionary PSO algorithm, namely CPSO, which divides a D-dimensional problem into k s-dimensional subproblems for some $s \ll D$. Shi et al. [17] adopted differential evolution (DE) [18] into CC, with decision variables split into two equal-sized subcomponents. Obviously, this decomposition strategy would not perform well on the problems with a very high dimensionality.

Yang et al. [13] proposed a random variable grouping method and applied it to CC. Unlike CPSO which relies on a fixed variable grouping from the start to the end of optimization, the random grouping method proposed by Yang et al. randomly shuffles all the decision variables into k s-dimensional subcomponents in each co-evolutionary cycle. It is shown in [13] that this random grouping strategy is effective in grouping two interdependent variables into one subcomponent for several cycles. The DE algorithm with this random grouping strategy, namely DECC-G, performs well on a set of large-scale optimization problems with up to 1000 dimensions [13].

The aforementioned grouping strategies use a pre-specified and fixed subcomponent size for decomposition. Therefore, a user needs to specify a value for either k or s before using these decomposition strategies, which may be difficult in practice. In addition, the performance of CC can be highly dependent on these specified values.

Adapting the subcomponent size can potentially improve the performance of CC [19]. Yang et al. [20] proposed a multilevel cooperatively co-evolutionary (MLCC) algorithm. MLCC uses a set of possible values of s for decomposition instead of a fixed subcomponent size. The performance of each subcomponent size used during optimization is measured according to the improvement of the best overall objective value. The subcomponent size with better performance would be selected in the next co-evolutionary cycle with a higher probability. Further enhancing the CCPSO algorithm [21] with an improved random variable grouping strategy, Li and Yao [22] proposed CCPSO2 to solve a set of large-scale optimization problems with up to 2000 dimensions.

Random grouping is ineffective when the number of interdependent variables is greater than five [19]. It is shown in [23] that a non-random method, namely delta grouping, is superior to random grouping on most of the CEC2010's benchmark functions [24]. The delta grouping method uses the average difference of a certain variable during optimization to detect interdependent variables. The variables with similar difference values are considered to be possible interdependent variables. However, this assumption may not always hold. For example, the delta grouping method cannot perform well when there is more than one subcomponent [23].

A given problem may be decomposed in an automatic way without knowing in advance its underlying structure, as suggested in [25]. In the beginning of the co-evolutionary process, all the variables are optimized separately by different subpopulations. A counter is used in [25] to compute the probability of grouping two variables together. If two variables in a randomly chosen individual can improve the best individual further, the counter is increased. At the end of each co-evolutionary cycle, the two variables with the maximum counter are grouped together. The subpopulations corresponding to the two variables are merged into one subpopulation. The CC with variable interaction learning (CCVIL) algorithm proposed by Chen et al. [26] adopts a two-stage approach. In the first stage, CCVIL detects the interaction between variables as done in [25] to complete the decomposition. In the second stage, CCVIL optimizes these decomposed groups in the fashion of the traditional CC [7].

Tezuka et al. [27] proposed the linkage identification by nonlinearity check for real-coded GAs (LINC-R). If the difference of function values with respect to a variable is independent on the difference of function values with respect to another variable, the two variables are separable. Omidvar et al. [28] provided a theoretical study of LINC-R and proposed a new method for detecting interdependent variables, namely differential grouping (DG). DG can identify the interdependent variables with a high accuracy. It is shown in [28] that CC with DG performs well on a set of large-scale optimization problems with up to 1000 dimensions.

For separable decision variables, it is shown in [29] that optimizing each variable separately may not be the best way for solving large-scale optimization problems. A more efficient approach is to group the separable variables into several groups. However, it may be difficult to determine the optimal group size.

When dealing with the partially separable problems, it is possible that there is imbalance between the contributions of different subpopulations to the improvement of the overall objective value. The round-robin strategy in the classic CC is no longer effective in handling this sort of problems since it allocates an equal amount of computational resources to each subpopulation, without considering the unequal contributions of the subpopulations. To overcome this problem, a contribution-based CC (CBCC) was proposed in [9] to allocate computational resources among the subpopulations based on their contributions to the improvement of the best overall objective value. CBCC emphasizes the contributions in the early stage of the evolutionary process. As a result, it may allocate most computational resources to the subpopulation whose initial contribution is greater but then drops after some generations. For the two variants of CBCC (CBCC1 and CBCC2), the experimental results in [30] show that CBCC1 is much less sensitive to the decomposition accuracy and the imbalance between the contributions of subpopulations than CBCC2. CBCC1 and CBCC2 are unable to adaptively respond to the dynamic contributions of subpopulations during optimization.

III. The proposed CC Framework

A new cooperatively co-evolutionary framework (CCFR) is presented in this section. CCFR aims at allocating computational resources intelligently among subpopulations according to the dynamic contributions of subpopulations to the improvement of the best overall objective value. Note that, CCFR adopts a two-stage approach similar to DECC-DG [28]. In the first stage, the decomposition is formed using a decomposition method; in the second stage, the resulting groups are optimized separately while the decomposition is kept fixed.

A. Saving Computation on Stagnant Subpopulations

CC makes subpopulations evolve using an evolutionary optimizer in a round-robin fashion. For the subcomponents that are easy to optimize, a small number of generations are enough for the corresponding subpopulations to enter a stagnant state, where these subpopulations do not make contributions to the improvement of the best overall objective value. In such a case, no computational resources would be allocated to these stagnant subpopulations. This will allow the CC algorithms to save some computational cost.

Suppose C_i denotes the i-th subcomponent after decomposition. For the subpopulation corresponding to C_i at the G-th generation, in order to check whether the subpopulation is stagnant, the mean and standard deviation of individuals' gene values in dimension j ($j \in C_i$) can be calculated as follows:

$$m_{j,G} = \frac{1}{N}\sum_{t=1}^{N} x_{t,j,G}, \tag{1}$$

$$std_{j,G} = \sqrt{\frac{1}{N}\sum_{t=1}^{N} (x_{t,j,G} - m_{j,G})^2}, \tag{2}$$

where N is the subpopulation size and $x_{t,j,G}$ is the j-th gene value of individual $\mathbf{x}_{t,G}$. $\mathbf{x}_{t,G} = (x_{t,1,G}, ..., x_{t,D,G})$. If the distribution of a population, i.e., the mean and standard deviation of individuals' gene values in dimension j, remains unchanged for several successive generations, this population is considered to be stagnant in this dimension [31]. Based on this strategy, we propose the following method for checking whether a subpopulation is stagnant in all dimensions.

$$\beta_{j,G} = \begin{cases} 1 & \text{if } m_{j,G} = m_{j,G-1} \text{ and} \\ & std_{j,G} = std_{j,G-1} \quad (3a) \\ 0 & \text{otherwise,} \quad (3b) \end{cases}$$

where $\beta_{j,G}$ denotes whether the values of $m_{j,G}$ and $std_{j,G}$ remain unchanged from the last generation in dimension j, and note that $\beta_{j,0} = 0$. γ_G denotes the number of dimensions where $\beta_{j,G} = 1$:

$$\gamma_G = \sum_{j \in C_i} \beta_{j,G}. \tag{4}$$

If the subpopulation does not change (i.e., no better individuals are generated), $\gamma_G = D_i$, where D_i is the dimensionality of subcomponent C_i. η_G denotes the number of successive generations where $\gamma_G = D_i$:

$$\eta_G = \begin{cases} \eta_{G-1} + 1 & \text{if } \gamma_G = D_i \quad (5a) \\ 0 & \text{otherwise,} \quad (5b) \end{cases}$$

and note that $\eta_0 = 0$. ρ_G is a flag to denote whether the subpopulation is stagnant at the G-th generation, and the value of ρ_G is calculated as follows:

$$\rho_G = \begin{cases} 1 & \text{if } \eta_G \geq U \quad (6a) \\ 0 & \text{otherwise,} \quad (6b) \end{cases}$$

where U is an integer with the value equal to D_i. Our experimental results show that the larger the subcomponent size is, the more generations its corresponding subpopulation takes to enter a stagnant state. According to the sensitivity study of U (provided in Sect. I in the supplementary material listed in the appendix), we use $U = D_i$. If the distribution of a subpopulation remains unchanged for several successive generations (i.e., $\eta_G \geq U$), ρ_G is set to one to indicate that the subpopulation is likely to stop evolution.

Some existing methods consider a population to be stagnant if the improvement of the best fitness value [32], [33] or the difference between the individuals [34], [35] is very small, even though the population still slowly converges to an optimum. Guo et al. [36], [37] considered an individual to be stagnant when the individual's fitness cannot be improved over several successive generations. This method is ineffective for problems with a plateau fitness landscape (e.g., the *Step* function [38]), where the fitness value of an individual does not change, while the values of the individual's decision variables change. Yang et al. [39] considered a population to be stagnant when the average distance among the individuals remains unchanged for several successive generations. However, it is possible that the distribution of the entire population changes (e.g., all the individuals vary with the same shift). In such a case, Yang's method may incorrectly classify the population as a stagnant one. Compared with the above stagnation detection methods, our proposed method is more accurate in identifying a stagnant population according to the mean and standard deviation of individuals' gene values.

For the subpopulations where $\rho_G = 1$, we exclude them from the co-evolutionary cycles, which means the stagnant subpopulations will not undergo evolution in the subsequent co-evolutionary cycles.

B. Resource Allocation Based on Contribution

The probability matching (PM) algorithm [40] and the adaptive pursuit (AP) algorithm [40] learn the optimal resource allocation among operators. These probability-based methods would allocate resources to the ineffective operators with a minimum probability. Based on the upper confidence bound (UCB) algorithm [41], Li et al. [42] proposed a method for allocating resources among operators, where the operator with the maximum relative fitness improvement is selected to take part in the evolutionary process [43], [44]. These methods based on relative fitness improvements allocate resources to the items (e.g., the converging items) whose absolute fitness improvements are very small but their relative fitness improvements are relatively large. In [45], the average absolute fitness improvements are used in determining resource allocation. Rainville et al. [46] proposed a resource allocation for CC based on binary rewards. A subpopulation is assigned a reward of one if the overall objective value becomes better, and zero otherwise. However, the binary rewards cannot reflect the real magnitudes of the improvements of the objective value. In this section, we propose a resource allocation strategy for CC based on the absolute improvements of the best overall objective value. Unlike the average absolute improvements in [45], our proposed method gives more consideration of resource allocation to the recent improvements of the overall objective value.

For a subpopulation (P_i), when P_i finishes evolution in a cycle, we calculate its contribution according to the improvement of the best overall objective value:

$$\Delta F_i = \frac{\Delta \hat{F}_i + \left| f(\hat{\mathbf{x}}_{best}) - f(\mathbf{x}_{best}) \right|}{2}, \tag{7}$$

where $f(\hat{\mathbf{x}}_{best})$ and $f(\mathbf{x}_{best})$ are the best overall objective values before and after P_i undergoes evolution in this cycle, respectively, and $\Delta\hat{F}_i$ is the last contribution of P_i. The initial value of ΔF_i is zero. Eq. (7) smoothly updates ΔF_i by averaging the last contribution (i.e., $\Delta\hat{F}_i$) and the current contribution (i.e., $|f(\hat{\mathbf{x}}_{best}) - f(\mathbf{x}_{best})|$) to the improvement of the best overall objective value. The more recent contribution $|f(\hat{\mathbf{x}}_{best}) - f(\mathbf{x}_{best})|$ is, the greater the effect of $|f(\hat{\mathbf{x}}_{best}) - f(\mathbf{x}_{best})|$ on the value of ΔF_i is. The effects of the early contributions on ΔF_i become smaller and smaller as the co-evolution progresses.

During the first co-evolutionary cycle, the subpopulations undergo evolution one by one. The values of ΔF_i for all the subpopulations are computed at the end of the first cycle.

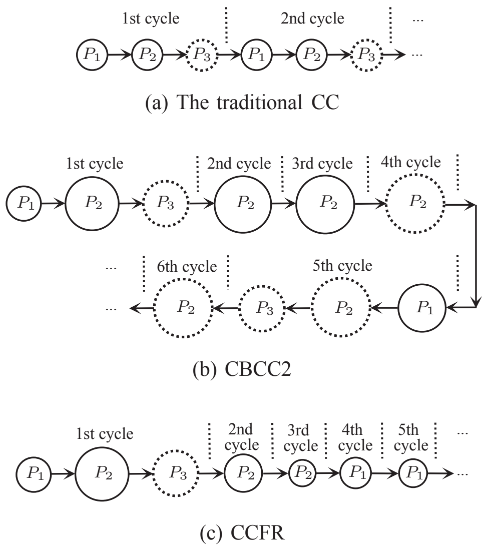


Fig. 1. The computational resource allocation in CC, CBCC2 and CCFR, where the circle size indicates the amount of contributions computed by the algorithms and the dotted circle indicates that the subpopulation is stagnant.

In the subsequent co-evolutionary cycles, we select only the subpopulation with the largest value of ΔF_i to undergo evolution. The value of ΔF_i is updated according to Eq. (7) at the end of each co-evolutionary cycle. The larger the value of ΔF_i is, the higher chance P_i has to undergo evolution in the future. If a subpopulation is stagnant according to Eq. (6), we set its contribution (ΔF_i) to zero. Therefore, the stagnant subpopulation will be excluded from the subsequent co-evolutionary cycles. When the values of ΔF_i are the same for all the subpopulations, we restart the process from the first co-evolutionary cycle. The advantage of doing so is that the subpopulation which is considered to be stagnant by mistake can resume its evolution. The above process is repeated until a termination criterion is met.

CBCC [9] can also allocate computational resources among the subpopulations according to their contributions to the improvement of the best overall objective value. The important difference between CCFR and CBCC is that CCFR responds faster to the recent changes of the overall objective value than CBCC. For CCFR, the contribution is updated smoothly by averaging the last and current contributions, whereas for CBCC, the contribution is accumulated from the beginning of the evolutionary process. Furthermore, CBCC does not take stagnant subpopulations into account.

Fig. 1 illustrates the computational resource allocation in the traditional CC [7], CBCC2 [9] (a variant of CBCC) and CCFR. The round-robin fashion in the traditional CC equally allocates computational resources among all the subpopulations without considering the different contributions of the subpopulations (see Fig. 1a). The traditional CC always allocates computational resources to stagnant subpopulations (e.g., P_3 in Fig. 1a), which is clearly wasteful. For CBCC, the contribution of each subpopulation is accumulated from the beginning of the evolutionary process, as shown in Fig. 1b, where different circle sizes suggest different amounts of the contributions of the subpopulations. CBCC2 allocates most computational

Algorithm 1 DECC [28]

```
   /*Suppose $C = \{C_1, \ldots, C_M\}$ is a decomposition and $P = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is a population.*/
1: $\mathbf{x}_{best} \leftarrow \operatorname{argmin}_{\mathbf{x} \in P} f(\mathbf{x})$;
2: for $k \leftarrow 1$ to $cycles$ do
3:     for $i \leftarrow 1$ to $M$ do
4:         $P_i \leftarrow \{x_{t,j} \mid x_{t,j} \in P, t = 1, \ldots, N, j \in C_i\}$;
5:         $P_i \leftarrow$Optimizer($\mathbf{x}_{best}, P_i, GEs$);
6:         $\{x_{t,j} \mid x_{t,j} \in P, t = 1, \ldots, N, j \in C_i\} \leftarrow P_i$;
7:         $\mathbf{x}_{best} \leftarrow \operatorname{argmin}_{\mathbf{x} \in P} f(\mathbf{x})$;
8:     end for
9: end for
```

resources to the subpopulation with the greatest accumulated contribution. In the second and third cycles, CBCC2 selects subpopulation P_2 with the greatest accumulated contribution to undergo evolution. From the second cycle, the contribution of P_2 in one cycle (i.e., the change of circle size) is small. Even in the case that P_2 has been stagnant, CBCC2 still deems P_2 makes the greatest contribution and allocates computational resources to P_2 (e.g., the sixth cycle in Fig. 1b). CBCC2 allocates computational resources to stagnant subpopulations P_2 and P_3. CCFR computes the contributions by averaging the last and current contributions at the end of each cycle. In Fig. 1c, it can be seen that for P_2, the circle size becomes smaller and smaller as the evolution progresses. The contribution that P_2 makes in the third cycle is relatively small. CCFR will select a subpopulation between P_1 and P_3 to undergo evolution in the next cycle. Although the last contribution of P_3 is greater than the one of P_1, CCFR selects P_1 to undergo evolution in the fourth cycle. This is because P_3 is stagnant and has been excluded from the cycles. The figure indicates that given an equal amount of computational resources, CCFR can potentially obtain better solutions than the traditional CC and CBCC2.

C. Obtaining the Best Overall Solution

In co-evolutionary cycles, many cooperatively co-evolutionary algorithms [9], [13], [20], [28] update the best overall solution to the original problem at the integrated-population level (see Step 7 in Algorithm 1). Take the following two-dimensional *Sphere* function as an example:

$$f(\mathbf{x}) = x_1^2 + x_2^2.$$

This function is additively separable [47]. Its ideal decomposition is $C = \{C_1, C_2\} = \{\{x_1\}, \{x_2\}\}$.

Suppose population P at a certain generation is as follows:

$$\begin{array}{r|c|c|} \cline{2-3} f{=}40 & \boldsymbol{6} & \boldsymbol{2} \\ \cline{2-3} f{=}58 & 7 & 3 \\ \cline{2-3} f{=}41 & 5 & 4 \\ \cline{2-3} \multicolumn{1}{r}{} & \multicolumn{2}{c}{P} \end{array},$$

where the current best overall solution $\mathbf{x}_{best} = (6,2)$ is shown in bold and italic font. Suppose that the evolutionary process (Steps 4 to 7 in Algorithm 1) for subpopulation P_1 is as follows:

$$\begin{array}{r|c|c|} \multicolumn{1}{r}{} & \multicolumn{1}{c}{P_1} & \multicolumn{1}{c}{} \\ \cline{2-3} f{=}40 & 6 & 2 \\ \cline{2-3} f{=}53 & 7 & 2 \\ \cline{2-3} f{=}29 & 5 & 2 \\ \cline{2-3} \end{array} \underset{\text{evolution}}{\rightarrow\cdots\rightarrow} \begin{array}{r|c|c|} \multicolumn{1}{r}{} & \multicolumn{1}{c}{P_1} & \multicolumn{1}{c}{} \\ \cline{2-3} f{=}20 & 4 & 2 \\ \cline{2-3} f{=}40 & 6 & 2 \\ \cline{2-3} f{=}13 & 3 & 2 \\ \cline{2-3} \end{array} \rightarrow \begin{array}{r|c|c|} \cline{2-3} f{=}20 & \boldsymbol{4} & \boldsymbol{2} \\ \cline{2-3} f{=}45 & 6 & 3 \\ \cline{2-3} f{=}25 & 3 & 4 \\ \cline{2-3} \multicolumn{1}{r}{} & \multicolumn{2}{c}{P} \end{array},$$

which produces an updated $\mathbf{x}_{best} = (4,2)$. Suppose that the evolutionary process for subpopulation P_2 is as follows:

$$\begin{array}{r|c|c|} \multicolumn{1}{r}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{P_2} \\ \cline{2-3} f{=}20 & 4 & 2 \\ \cline{2-3} f{=}25 & 4 & 3 \\ \cline{2-3} f{=}32 & 4 & 4 \\ \cline{2-3} \end{array} \underset{\text{evolution}}{\rightarrow\cdots\rightarrow} \begin{array}{r|c|c|} \multicolumn{1}{r}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{P_2} \\ \cline{2-3} f{=}17 & 4 & 1 \\ \cline{2-3} f{=}16 & 4 & 0 \\ \cline{2-3} f{=}25 & 4 & 3 \\ \cline{2-3} \end{array} \rightarrow \begin{array}{r|c|c|} \cline{2-3} f{=}17 & \boldsymbol{4} & \boldsymbol{1} \\ \cline{2-3} f{=}36 & 6 & 0 \\ \cline{2-3} f{=}18 & 3 & 3 \\ \cline{2-3} \multicolumn{1}{r}{} & \multicolumn{2}{c}{P} \end{array},$$

which similarly produces an updated $\mathbf{x}_{best} = (4,1)$. When this co-evolutionary cycle ends, all the possible combinations of the individuals from different subpopulations are (4,1), (4,0), (4,3), (6,1), (6,0), (6,3), (3,1), (3,0) and (3,3). Each combination is an overall solution to the problem. Among all the combinations, the best overall solution is (3,0). For a population with population size N and M subpopulations, the number of the combinations is N^M. We improve the CC framework in Algorithm 1 through updating $\mathbf{x}_{best}$ in the following way. In the case that the subcomponents corresponding to the M subpopulations are separable between each other, $\mathbf{x}_{best}$ obtained by the improved CC framework is the best overall solution from the N^M combinations.

According to the definition of separability [24], [47]:

$$\operatorname*{argmin}_{\mathbf{x}} f(\mathbf{x}) = \left(\operatorname*{argmin}_{\mathbf{x}_1} f(\mathbf{x}_1, \ldots), \ldots, \operatorname*{argmin}_{\mathbf{x}_M} f(\ldots, \mathbf{x}_M)\right), \qquad (8)$$

for a separable function $f(\mathbf{x})$ with M independent subcomponents, the following equation holds:

$$\operatorname*{argmin}_{\mathbf{x} \in Z} f(\mathbf{x}) = \left(\operatorname*{argmin}_{\mathbf{x}_1 \in P_1} f(\mathbf{x}_1, \ldots), \ldots, \operatorname*{argmin}_{\mathbf{x}_M \in P_M} f(\ldots, \mathbf{x}_M)\right), \qquad (9)$$

where Z is the set of all the possible combinations of the individuals from $P_1, \ldots, P_M$. Eq. (9) simply states that if the subcomponents are separable, the combination of the best solution from each subpopulation must be the best overall solution from Z to the original problem. When a decomposition is formed, we set the best overall solution $\mathbf{x}_{best}$ as follows:

$$\mathbf{x}_{best} = \left(\operatorname*{argmin}_{\mathbf{x}_1 \in P_1} f(\mathbf{x}_1, \mathbf{x}_{best}^{P_1}), \ldots, \operatorname*{argmin}_{\mathbf{x}_M \in P_M} f(\mathbf{x}_M, \mathbf{x}_{best}^{P_M})\right), \qquad (10)$$

where $\mathbf{x}_{best}^{P_i} = \{x \mid x \in \mathbf{x}_{best}, x \notin P_i\}$, which consists of $\mathbf{x}_{best}$ with the dimensions of P_i removed. $\mathbf{x}_{best}$ is a concatenation of all best solutions from the M subpopulations $(P_1, \ldots, P_M)$, as constructed in [48]. In Algorithm 1, Step 5 is changed as follows:

$$(P_i, \mathbf{x}_{best}) \leftarrow \text{Optimizer}(\mathbf{x}_{best}, P_i, GEs),$$

where $\mathbf{x}_{best}$ is updated at the end of the co-evolutionary process for each subpopulation, and Step 7 is removed. The above co-evolutionary example changes as follows:

$$\begin{array}{r|c|c|} \multicolumn{1}{r}{} & \multicolumn{1}{c}{P_1} & \multicolumn{1}{c}{} \\ \cline{2-3} f{=}40 & 6 & 2 \\ \cline{2-3} f{=}53 & 7 & 2 \\ \cline{2-3} f{=}29 & 5 & 2 \\ \cline{2-3} \end{array} \underset{\text{evolution}}{\rightarrow\cdots\rightarrow} \begin{array}{r|c|c|} \multicolumn{1}{r}{} & \multicolumn{1}{c}{P_1} & \multicolumn{1}{c}{} \\ \cline{2-3} f{=}20 & 4 & 2 \\ \cline{2-3} f{=}40 & 6 & 2 \\ \cline{2-3} f{=}13 & \boldsymbol{3} & \boldsymbol{2} \\ \cline{2-3} \end{array},$$

which produces an updated $\mathbf{x}_{best} = (3,2)$, and

$$\begin{array}{r|c|c|} \multicolumn{1}{r}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{P_2} \\ \cline{2-3} f{=}13 & 3 & 2 \\ \cline{2-3} f{=}18 & 3 & 3 \\ \cline{2-3} f{=}25 & 3 & 4 \\ \cline{2-3} \end{array} \underset{\text{evolution}}{\rightarrow\cdots\rightarrow} \begin{array}{r|c|c|} \multicolumn{1}{r}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{P_2} \\ \cline{2-3} f{=}10 & 3 & 1 \\ \cline{2-3} f{=}9 & \boldsymbol{3} & \boldsymbol{0} \\ \cline{2-3} f{=}18 & 3 & 3 \\ \cline{2-3} \end{array},$$

which similarly produces an updated $\mathbf{x}_{best} = (3,0)$. From the above evolutionary process, it can be seen that $\mathbf{x}_{best}$ is always updated as the best overall solution during evolution. Note that, if there is interdependence between subcomponents, $\mathbf{x}_{best}$

Algorithm 2 CCFR

```
1: Generate a decomposition $C = \{C_1, \ldots, C_M\}$;
2: Generate a uniform random population $P = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$;
3: Compute $\mathbf{x}_{best} \leftarrow \underset{\mathbf{x}\in P}{\operatorname{argmin}}\, f(\mathbf{x})$;
4: Set the value of $\mathbf{x}_{best}$ using Eq. (10);
5: $\Delta F_i \leftarrow 0, G_i \leftarrow 0, i = 1, 2, \ldots, M$;
6: while the termination criterion is not met do
7:     For each subpopulation, reset $\eta$ (see Eq. (5)) to 0;
8:     for $i \leftarrow 1$ to $M$ do
9:         $\hat{\mathbf{x}}_{best} \leftarrow \mathbf{x}_{best}$;
10:        $P_i \leftarrow \{x_{t,j} \mid x_{t,j} \in P, t = 1, \ldots, N, j \in C_i\}$;
11:        $(P_i, \mathbf{x}_{best}, \rho_{G_i}, G_i) \leftarrow \text{Optimizer}(\mathbf{x}_{best}, P_i, GEs, G_i)$;
12:        $\{x_{t,j} \mid x_{t,j} \in P, t = 1, \ldots, N, j \in C_i\} \leftarrow P_i$;
13:        $\Delta F_i \leftarrow (\Delta F_i + |f(\hat{\mathbf{x}}_{best}) - f(\mathbf{x}_{best})|)/2$;
14:        if $\rho_{G_i}$ equals 1 then
15:            $\Delta F_i \leftarrow 0$;
16:        end if
17:    end for
18:    while $\min(\Delta F_i | i = 1, \ldots, M) \neq \max(\Delta F_i | i = 1, \ldots, M)$ do
19:        $i \leftarrow$ the index of the maximum $\Delta F_i$;
20:        $\hat{\mathbf{x}}_{best} \leftarrow \mathbf{x}_{best}$;
21:        $P_i \leftarrow \{x_{t,j} \mid x_{t,j} \in P, t = 1, \ldots, N, j \in C_i\}$;
22:        $(P_i, \mathbf{x}_{best}, \rho_{G_i}, G_i) \leftarrow \text{Optimizer}(\mathbf{x}_{best}, P_i, GEs, G_i)$;
23:        $\{x_{t,j} \mid x_{t,j} \in P, t = 1, \ldots, N, j \in C_i\} \leftarrow P_i$;
24:        $\Delta F_i \leftarrow (\Delta F_i + |f(\hat{\mathbf{x}}_{best}) - f(\mathbf{x}_{best})|)/2$;
25:        if $\rho_{G_i}$ equals 1 then
26:            $\Delta F_i \leftarrow 0$;
27:        end if
28:    end while
29: end while
```

Algorithm 3 $(P_i, \mathbf{x}_{best}, \rho_G, G) \leftarrow \text{Optimizer}(\mathbf{x}_{best}, P_i, GEs, G_0)$

```
1: $G \leftarrow G_0$;
2: For $\mathbf{x} \in P_i$, evaluate $(\mathbf{x}, \mathbf{x}_{best}^{P_i})$;
3: while $G < G_0 + GEs$ do
4:     for $\mathbf{x} \in P_i$ do
5:         $\hat{\mathbf{x}} \leftarrow \text{Reproduction}(\mathbf{x})$; /*evolutionary process*/
6:         Evaluate $(\hat{\mathbf{x}}, \mathbf{x}_{best}^{P_i})$;
7:         if $(\hat{\mathbf{x}}, \mathbf{x}_{best}^{P_i})$ is better than $(\mathbf{x}, \mathbf{x}_{best}^{P_i})$ then
8:             $\mathbf{x} \leftarrow \hat{\mathbf{x}}$;
9:         end if
10:        if $(\hat{\mathbf{x}}, \mathbf{x}_{best}^{P_i})$ is better than $\mathbf{x}_{best}$ then
11:            $\mathbf{x}_{best} \leftarrow (\hat{\mathbf{x}}, \mathbf{x}_{best}^{P_i})$;
12:        end if
13:    end for
14:    $G \leftarrow G + 1$;
15:    Compute $\rho_G$ using Eq. (6);
16:    if $\rho_G$ equals 1 then
17:        Terminate the algorithm and return;
18:    end if
19: end while
```

obtained by the above changed evolutionary process may not be the best overall solution.

D. CCFR

Algorithm 2 summarizes the proposed CCFR. Steps 8 to 17 compute the contribution (i.e., the value of ΔF_i) of each subpopulation. Steps 18 to 28 select the subpopulation with the greatest contribution to undergo evolution and update its contribution when the evolution ends. When all the subpopulations make an equal contribution, CCFR goes to Step 8 to reset the contribution of each subpopulation. The above process is repeated until a termination criterion is met. Steps 11 and 22 invoke the evolutionary process for a subpopulation, which is shown in Algorithm 3.

In Algorithm 3, a subpopulation undergoes evolution for a pre-specified number of generations, i.e., GEs. Steps 15 to 18 check whether a subpopulation is stagnant. If the subpopulation is identified as a stagnant one, CCFR will stop the subpopulation evolving. In Algorithm 3, the best overall solution $\mathbf{x}_{best}$ is updated when a better solution is found. In the end, $\mathbf{x}_{best}$ is returned to Algorithm 2.

Compared with the traditional CC, CCFR needs extra computation to initialize the best overall solution before the co-evolutionary cycles begin (Step 4 in Algorithm 2), and the computational complexity is $O(M \cdot N)$. CCFR also needs extra computation to check whether a subpopulation is stagnant at each generation (Step 15 in Algorithm 3), and the computational complexity is $O(D_i \cdot N)$.

IV. Experimental Studies

A set of 35 test instances with 1000 dimensions proposed in the IEEE CEC'2010 and CEC'2013 special sessions on large-scale global optimization were used to study the performance of CCFR. The detailed description of these test instances is given in [24], [47]. Compared with the CEC'2010 functions, the CEC'2013 functions have four new characteristics: nonuniform subcomponent sizes, imbalance in the contributions of subcomponents, functions with overlapping subcomponents, and new transformations to the base functions.

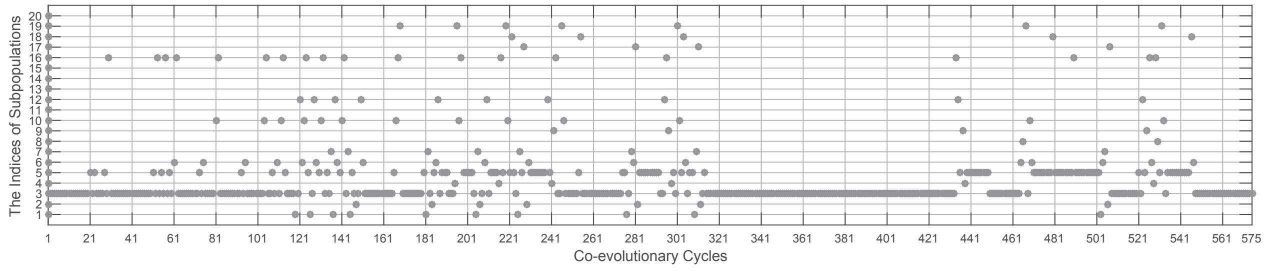
In the experimental studies, CCFR is compared with seven CC algorithms (DECC-G [13], MLCC [49], DECC-D [23], DECC-DML [23], DECC [28], CBCC1 [9] and CBCC2 [9]) and two memetic algorithms (MA-SW-Chains [50] and MOS-CEC2013 [51]). The two memetic algorithms were ranked the first in the IEEE CEC'2010 and CEC'2013 competitions on large-scale global optimization, respectively. We set the maximum number of fitness evaluations to $MaxFEs = 3 \times 10^6$ as the termination criterion, as suggested in [24]. For the competitors of CCFR, the parameters were set to the values as used in their publications. To make a fair comparison, CCFR and the other CC algorithms under comparison adopt the same settings of the following parameters.

1) The subcomponent optimizer is SaNSDE [52], a variant of differential evolution (DE) [18]. The population size of SaNSDE was set to 50.
2) The pre-specified number of the evolutionary generations, i.e., GEs in Algorithm 3, was set to 100.

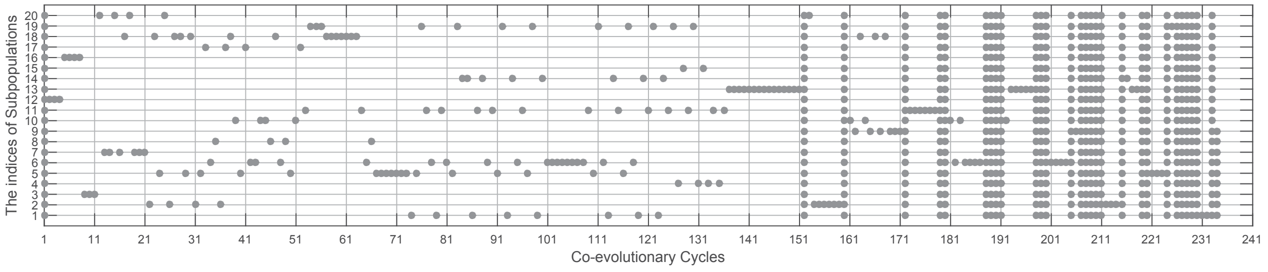
A. Behavior of CCFR

In this section, the behavior of CCFR is studied. The grouping of variables is an ideal decomposition, which was done manually using the prior knowledge of the benchmark functions.

Fig. 2 shows the activation of the subpopulations in a single run on two CEC'2013 functions (f_8 and f_{10}), which have 20 separable subcomponents. The contributions of all the subpopulations were computed in the first co-evolutionary cycle. For f_8, because the third subcomponent has the largest weight value [47], the corresponding subpopulation (i.e., P_3) has the largest contribution to the improvement of the best overall objective value. In Fig. 2a, it can be seen that after

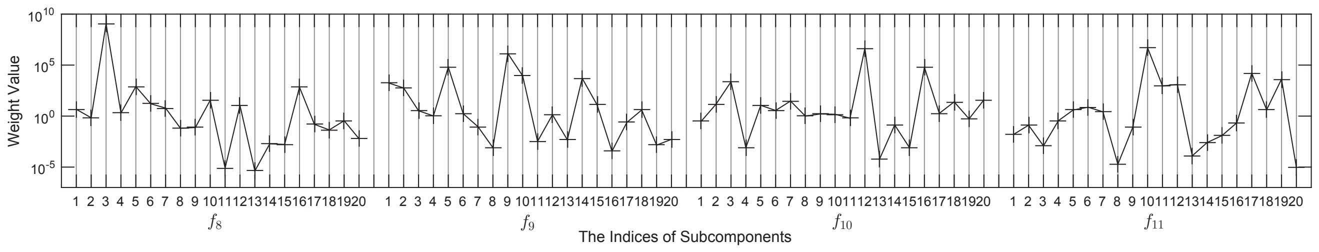


(a) f_8

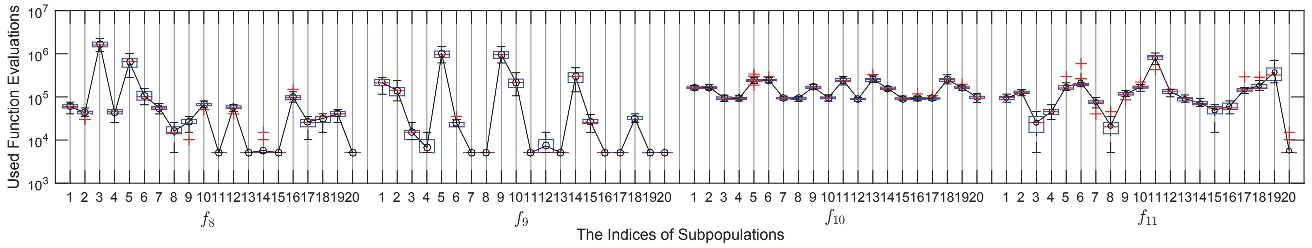


(b) f_{10}

Fig. 2. The activation of subpopulations for CCFR-I on two selected CEC'2013 functions in a single run, where the filled circle point indicates that the subpopulation undergoes evolution in the corresponding co-evolutionary cycle.



(a) The weight values of subcomponents



(b) The box plot of the function evaluations used by each subpopulation to optimize its corresponding subcomponent over 25 independent runs, where the circle point indicates the mean number of function evaluations used by each subpopulation over 25 independent runs.

Fig. 3. The computational resource allocation among the subpopulations in CCFR-I on four selected CEC'2013 functions.

the first cycle, P_3 underwent evolution in the subsequent successive cycles. The contribution of P_3 became smaller and smaller as the evolution progressed. In the 21st cycle, P_5, whose corresponding subcomponent has the second largest weight value, underwent evolution. From Fig. 2a, two observations can be made: 1) the subpopulations undergo evolution alternately; 2) most computational resources are spent on P_3 and P_5, whose corresponding subcomponents have the largest and second largest weight values, respectively. For f_8, according to the dynamic contributions of the subpopulations, CCFR can adaptively allocate computational resources among the subpopulations.

For f_{10}, it can be seen in Fig. 2b that P_{12}, whose corresponding subcomponent has the largest weight value [47], underwent evolution in several successive cycles. Because SaNSDE, the optimizer used by CCFR, was not able to solve this function effectively, P_{12} was stagnant. The distribution of P_{12} remained unchanged for several successive generations. In the fourth cycle, CCFR considered P_{12} to be stagnant according to Eq. (6) and excluded it from the subsequent cycles. In the 152nd cycle, all the subpopulations were stagnant. The co-evolution restarted from the first cycle. All the subpopulations underwent evolution one by one.

Fig. 3 shows the resource allocation in CCFR-I on four CEC'2013 functions (f_8–f_{11}), which have 20 separable subcomponents. The weight values of the subcomponents are significantly different (see Fig. 3a), which results in the significantly different contributions of the subpopulations to

TABLE I
THE AVERAGE FITNESS VALUES ± STANDARD DEVIATIONS ON THE CEC'2010 AND CEC'2013 FUNCTIONS OVER 25 INDEPENDENT RUNS. THE SIGNIFICANTLY BETTER RESULTS ARE IN BOLD FONT (WILCOXON RANK SUM TEST WITH HOLM p-VALUE CORRECTION, α=0.05). R^+, R^- AND p-VALUE ARE OBTAINED THROUGH MULTIPLE-PROBLEM ANALYSIS BY THE WILCOXON TEST BETWEEN CCFR-I AND ITS COMPETITORS.

CEC'2010 Functions				
F	CCFR-I	CBCC1-I	CBCC2-I	CC-I
f_1	**1.2e-05±4.9e-06**	9.9e+06±1.3e+07↑	9.9e+06±1.3e+07↑	3.5e+11±2.0e+10↑
f_2	**2.7e+01±5.2e+00**	4.7e+03±4.8e+02↑	4.7e+03±4.8e+02↑	9.4e+03±2.1e+02↑
f_3	**4.6e+00±4.6e-01**	1.2e+01±3.7e-01↑	1.2e+01±3.7e-01↑	2.0e+01±4.4e-02↑
f_4	8.3e+10±6.2e+10	6.0e+10±4.4e+10	9.9e+10±2.7e+10↑	3.4e+14±7.5e+13↑
f_5	7.2e+07±1.3e+07	6.8e+07±1.0e+07	6.7e+07±9.1e+06	4.9e+08±2.4e+07↑
f_6	**7.7e+05±7.1e+05**	1.3e+06±6.4e+05↑	1.3e+06±6.8e+05↑	1.1e+07±7.5e+05↑
f_7	**1.5e-03±2.5e-04**	5.9e+04±9.3e+03↑	8.4e+04±1.9e+04↑	7.7e+10±9.6e+09↑
f_8	**3.2e+05±1.1e+06**	8.6e+05±1.6e+06↑	1.0e+06±1.7e+06↑	1.8e+14±9.3e+13↑
f_9	9.4e+06±1.2e+06	1.7e+07±2.1e+07	2.8e+09±1.8e+09↑	9.4e+08±7.1e+07↑
f_{10}	**1.4e+03±1.0e+02**	3.0e+03±1.7e+02↑	4.5e+03±6.6e+02↑	4.8e+03±6.7e+01↑
f_{11}	**1.0e+01±2.7e+00**	2.2e+01±3.2e+00↑	2.4e+01±2.7e+00↑	4.1e+01±1.5e+00↑
f_{12}	**1.2e+00±4.6e+00**	1.8e+04±6.5e+03↑	2.5e+04±7.3e+03↑	4.9e+05±3.4e+04↑
f_{13}	**3.2e+02±9.9e+01**	1.9e+04±6.3e+03↑	2.8e+04±5.4e+03↑	1.5e+07±4.1e+06↑
f_{14}	**2.5e+07±2.9e+06**	2.8e+07±2.1e+06↑	9.5e+09±5.2e+08↑	2.7e+07±2.1e+06↑
f_{15}	**2.8e+03±1.3e+02**	4.0e+03±1.5e+02↑	4.2e+03±1.6e+02↑	4.0e+03±1.6e+02↑
f_{16}	2.0e+01±2.6e+00	1.9e+01±3.2e+00	2.0e+01±3.4e+00	2.0e+01±4.0e+00
f_{17}	**9.8e+00±1.1e+01**	3.5e+01±4.9e+01↑	1.4e+02±4.4e+01↑	2.2e+01±3.7e+01↑
f_{18}	1.1e+03±1.8e+02	1.1e+03±1.8e+02	1.4e+03±1.9e+02↑	1.0e+03±1.7e+02
f_{19}	1.2e+06±9.5e+04	1.2e+06±9.5e+04	1.2e+06±9.5e+04	1.2e+06±9.5e+04
f_{20}	1.0e+09±9.0e+08	1.0e+09±9.0e+08	1.0e+09±9.0e+08	1.0e+09±9.0e+08
R^+	—	167.0	194.0	204.0
R^-	—	43.0	16.0	6.0
p-value	—	2.06e-02	8.92e-04	2.19e-04
CEC'2013 Functions				
F	CCFR-I	CBCC1-I	CBCC2-I	CC-I
f_1	**1.3e-05±3.2e-06**	1.4e+07±3.6e+07↑	1.4e+07±3.6e+07↑	3.7e+11±1.5e+10↑
f_2	**5.5e-01±1.5e+00**	2.1e+04±9.9e+02↑	2.1e+04±9.9e+02↑	8.5e+04±5.1e+03↑
f_3	**2.0e+01±3.1e-07**	2.1e+01±1.1e-02↑	2.1e+01±1.1e-02↑	2.1e+01±9.1e-03↑
f_4	**4.5e+07±1.7e+07**	1.6e+08±6.0e+07↑	6.6e+10±5.6e+09↑	1.7e+12±4.8e+11↑
f_5	2.5e+06±2.7e+05	2.5e+06±4.2e+05	2.4e+06±4.5e+05	1.2e+07±6.9e+05↑
f_6	1.1e+06±1.2e+03	1.1e+06±1.9e+03↓	1.1e+06±1.7e+03↓	1.1e+06±1.6e+03↑
f_7	**8.6e+06±1.9e+07**	1.9e+07±2.4e+07↑	9.6e+07±3.7e+08↑	4.2e+09±1.1e+09↑
f_8	**9.6e+09±1.6e+10**	2.0e+13±2.8e+13↑	1.0e+12±1.3e+11↑	4.7e+13±2.8e+13↑
f_9	**1.9e+08±2.8e+07**	2.5e+08±3.8e+07↑	2.2e+08±2.8e+07↑	2.9e+08±5.2e+07↑
f_{10}	9.5e+07±1.9e+05	9.4e+07±2.8e+05↓	9.4e+07±2.3e+05↓	9.4e+07±2.9e+05↓
f_{11}	**3.3e+08±3.2e+08**	3.0e+09±1.0e+10↑	4.9e+10±9.5e+10↑	2.2e+09±8.4e+09↑
f_{12}	6.0e+08±7.1e+08	6.1e+08±7.1e+08	6.1e+08±7.1e+08	6.1e+08±7.1e+08
f_{13}	9.3e+08±5.3e+08	9.5e+08±5.4e+08	9.5e+08±5.4e+08	9.5e+08±5.4e+08
f_{14}	2.1e+09±2.1e+09	2.2e+09±2.1e+09	2.2e+09±2.1e+09	2.2e+09±2.1e+09
f_{15}	8.2e+06±3.3e+06	8.3e+06±3.3e+06	8.3e+06±3.3e+06	8.3e+06±3.3e+06
R^+	—	109.0	107.0	115.0
R^-	—	11.0	13.0	5.0
p-value	—	3.36e-03	5.37e-03	6.10e-04

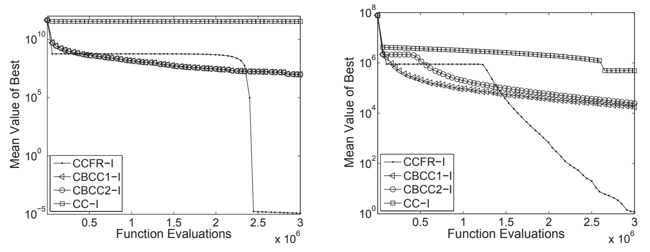
The symbols ↑ and ↓ denote that the CCFR-I algorithm performs significantly better than and worse than this algorithm by the Wilcoxon rank sum test at the significance level of 0.05, respectively.

the improvement of the best overall objective value. It can be seen in Fig 3 that for f_8–f_{11} except f_{10}, the larger the weight value of a subcomponent is, the more resources its corresponding subpopulation uses for evolution. As discussed before, the optimizer used in CCFR was not able to solve f_{10} effectively, so all the subpopulations were stagnant after some cycles. All the subpopulations then underwent evolution one by one. Therefore, for f_{10}, the computational resources allocated to different subpopulations do not differ greatly (see Fig. 3b).

B. Comparison Between CCFR and Other CC Frameworks

In this section, CCFR is compared with two variants of CBCC (CBCC1 and CBCC2) [9] and the traditional CC [7]. The grouping of variables for CCFR-I, CBCC-I and CC-I is an ideal decomposition, which was done manually using the prior knowledge of the functions. All the function evaluations are used for optimization. For the separable variables, CCFR and CC optimize the variables separately, while CBCC optimizes the variables together [28]. The only difference between CCFR-I, CBCC-I and CC-I is the cooperatively co-evolutionary frameworks they employ. Table I summarizes the results of CCFR-I, CBCC1-I, CBCC2-I and CC-I.

1) Comparison on the IEEE CEC'2010 Functions: The results show that CCFR-I performs significantly better than the other peer algorithms on 13 out of 20 functions. CCFR-I outperforms the other peer algorithms on all the separable functions (f_1–f_3) and most of the partially separable functions (f_4–f_{18}). For the partially separable functions on which CCFR-I performs worse, the differences between the results of CCFR-I and the other peer algorithms are not significant. For the functions on which CCFR-I performs better, the differences are significant, especially for f_7, f_{12}, f_{13} and f_{17}. For the nonseparable functions (f_{19} and f_{20}), all the variables are grouped into one subcomponent and are optimized together, hence there is no issue of computational resource allocation. CCFR-I, CBCC-I and CC-I have similar performance on the nonseparable functions.



(a) f_1 (b) f_{12}

Fig. 4. The average convergence on two selected CEC'2010 functions over 25 independent runs.

Fig. 4 shows the convergence of four CC algorithms. f_1 is a fully separable function in which each variable has a weight value. These weight values grow as the indices of the variables increase. f_{12} is a partially separable function with 10 nonseparable subcomponents and 500 separable variables.

CC cannot save computational resources on stagnant subpopulations. As can be seen in Fig. 4, the convergence speed of CC-I is very slow. In contrast, CCFR can stop stagnant subpopulations from evolving. As a result, CCFR spends much less computational resources on the separable variables and converges faster than CC-I. In the beginning of the evolutionary process, CCFR-I converges slowly. This is because CCFR-I optimizes all the subcomponents including the separable variables one by one in the first co-evolutionary cycle. When the first cycle ends (about 2.5×10^6 function evaluations for f_1; about 1.3×10^6 function evaluations for f_{12}), CCFR-I starts to select the subpopulation with the greatest contribution to undergo evolution, hence the convergence speed of CCFR-I increases. CBCC groups all the separable variables into one subcomponent and all the separable variables are optimized together [28], which loses the power of the divide-and-conquer strategy of CC. In Fig. 4a, it can be seen that CBCC1-I and CBCC2-I converge slowly on f_1. The best overall objective value of f_1 drops sharply when CCFR-I completes the first co-evolutionary cycle. For f_{12}, CCFR-I converges faster than CBCC1-I and CBCC2-I (see Fig. 4b). CBCC

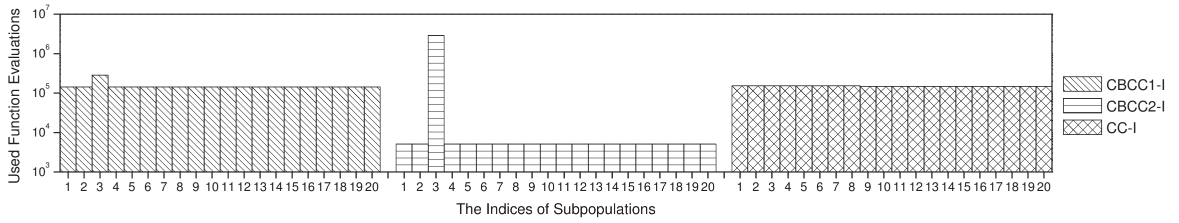


Fig. 5. The average function evaluations used by each subpopulation to optimize its corresponding subcomponent on a CEC'2013 function (f_8) over 25 independent runs.

allocates computational resources among subpopulations according to the accumulated contributions. Emphasizing the recent contributions, CCFR adapts the computational resource allocation to the real-time contributions of subpopulations better than CBCC. The experimental results in a single run on f_{12} showed that for the third subpopulation, CBCC1-I and CBCC2-I used about 5×10^5 and 1×10^6 function evaluations to improve the best overall objective value by 6.9×10^5. CCFR-I used about 1.9×10^5 function evaluations to make the improvement of 6.9×10^5. When the real-time contribution of the third subpopulation was relatively small, CBCC still allocated computational resources to the subpopulation, while CCFR allocated resources to some other subpopulation which made a relatively greater real-time contribution.

2) Comparison on the IEEE CEC'2013 Functions: To further investigate the effect of imbalance, CCFR-I, CBCC-I and CC-I were tested on the CEC'2013 functions. The results show that CCFR-I significantly outperforms the other peer algorithms on 8 out of 15 functions. CCFR-I performs significantly better than the other peer algorithms on all the separable functions (f_1–f_3) and most of the partially separable functions (f_4–f_{11}). CCFR-I, CBCC-I and CC-I have similar performance on nonseparable functions f_{12}–f_{15}. For the partially separable functions on which CCFR-I performs worse, the differences between the results of CCFR-I and the other peer algorithms are not significant. For the functions on which CCFR-I performs better, the differences are significant, especially for f_4, f_7, f_8 and f_{11}, where CCFR-I outperforms the other peer algorithms by several orders of magnitude.

Fig. 5 shows the average function evaluations used by each subpopulation to optimize its corresponding subcomponent for CBCC1-I, CBCC2-I and CC-I on f_8 over 25 independent runs. For f_8, the weight values of the subcomponents are significantly different (see Fig. 3a). It can be seen in Fig. 5 that CC-I allocates equal computational resources to all the subpopulations. CBCC1-I and CBCC2-I allocate equal computational resources to all the subpopulations except the third one (P_3). In the beginning of the evolutionary process, P_3 makes the greatest contribution. Therefore, CBCC1-I and CBCC2-I allocate more computational resources to P_3. In the subsequent co-evolutionary cycles, the contribution of P_3 in one cycle drops, but CBCC1-I and CBCC2-I still deem P_3 makes the greatest contribution and allocate resources to P_3 rather than some other subpopulation which makes the greatest real-time contribution. In contrast, CCFR-I allocates computational resources to P_5 with the greatest real-time contribution when the real-time contribution of P_3 is small.

TABLE II
AVERAGE RANKINGS ON THE CEC'2010 AND CEC'2013 FUNCTIONS (FRIEDMAN TEST). THE BEST RESULT IS IN BOLD FONT.

	CCFR-I	CBCC1-I	CBCC2-I	CC-I	p-value
Average Ranking	**1.4000**	2.3714	2.8286	3.4000	1.15e-10

TABLE III
THE AVERAGE FITNESS VALUES ± STANDARD DEVIATIONS ON FOUR PARTIALLY SEPARABLE CEC'2013 FUNCTIONS (f_8–f_{11}) OVER 25 INDEPENDENT RUNS. THE SIGNIFICANTLY BETTER RESULTS ARE IN BOLD FONT (WILCOXON RANK SUM TEST WITH HOLM p-VALUE CORRECTION, α=0.05). R^+, R^- AND p-VALUE HAVE SIMILAR MEANINGS AS IN TABLE I.

F	CCFR-I	ICBCC1-I	ICBCC2-I	ICC-I
f_8	**9.6e+09±1.6e+10**	1.9e+13±2.7e+13↑	9.9e+11±1.3e+11↑	4.7e+13±2.6e+13↑
f_9	**1.9e+08±2.8e+07**	2.5e+08±3.8e+07↑	2.2e+08±2.9e+07↑	2.8e+08±5.4e+07↑
f_{10}	9.5e+07±1.9e+05	9.5e+07±2.8e+05↓	9.5e+07±3.1e+05↓	9.5e+07±2.8e+05
f_{11}	3.3e+08±3.2e+08	5.2e+08±4.6e+08	7.9e+09±1.2e+10↑	1.8e+09±6.1e+09↑
R^+	—	9.0	9.0	9.0
R^-	—	1.0	1.0	1.0
p-value	—	2.50e-01	2.50e-01	2.50e-01

The symbols ↑ and ↓ have similar meanings as in Table I.

Allocating more computational resources to the subpopulation with the greatest contribution increases the probability of making a greater improvement of the best overall objective value. In short, for f_8, the result of CCFR-I is significantly better than those of CBCC1-I, CBCC2-I and CC-I (see Table I).

The average ranking of CCFR-I is the best among the four CC algorithms on the CEC'2010 and CEC'2013 functions (see Table II). The results in this section show that CCFR can make better use of computational resources than CBCC and CC on both the CEC'2010 and CEC'2013 functions.

In order to show the effect of the contribution-based resource allocation (see Sect. III-B) on the overall performance of CCFR, we compared CCFR-I with ICBCC2-I, ICBCC1-I and ICC-I, which are the improved CBCC1-I, CBCC2-I and CC-I, respectively. ICBCC2-I, ICBCC1-I and ICC-I adopt the components of CCFR (see Sect. III-A to Sect. III-C) except the contribution-based resource allocation. Table III summarizes the results on partially separable CEC'2013 functions f_8–f_{11}. The comparison between the results in Table I and Table III shows that the components of CCFR proposed in Sect. III-A and Sect. III-C improve the performance of CBCC1-I, CBCC2-I and CC-I on the four CEC'2013 functions (f_8–f_{11}). However, CCFR-I still outperforms the other CC algorithms on most of the four functions due to its better contribution-based resource allocation.

The scalability study of CCFR-I on the block-rotated ellip-

TABLE IV
THE AVERAGE FITNESS VALUES ± STANDARD DEVIATIONS ON THE CEC'2010 AND CEC'2013 FUNCTIONS OVER 25 INDEPENDENT RUNS. THE SIGNIFICANTLY BETTER RESULTS ARE IN BOLD FONT (WILCOXON RANK SUM TEST WITH HOLM p-VALUE CORRECTION, α=0.05). R^+, R^- AND p-VALUE HAVE SIMILAR MEANINGS AS IN TABLE I.

CEC'2010 Functions

F	CCFR-IDG2	DECC-G	MLCC	DECC-D	DECC-DML
f_1	2e-5±7e-6	4e-7±1e-7↓	8e-7±4e-7↓	**1e-22±9e-21**↓	3e-7±9e-7↓
f_2	1.7e2±9e0	1.3e3±3e1↑	**3e-3±5e-3**↓	6.5e1±4e1↓	1.0e1±2e1↓
f_3	1.2e1±4e-1	1.1e0±4e-1↓	**1e-2±3e-2**↓	2.3e0±2e-1↓	3e-1±7e-1↓
f_4	**1e11±8e10**	2e13±5e12↑	1e14±4e13↑	3e12±9e11↑	1e14±2e14↑
f_5	**9.2e7±2e7**	2.5e8±7e7↑	5.0e8±1e8↑	2.9e8±1e8↑	5.0e8±1e8↑
f_6	**6.8e5±7e5**	5.3e6±1e6↑	1.9e7±2e6↑	5.9e6±5e6↑	1.7e7±6e6↑
f_7	**2e-3±3e-4**	8.1e8±5e8↑	5e10±2e10↑	1.5e5±2e5↑	3e10±5e10↑
f_8	**3.2e5±1e6**	6.8e7±3e7↑	8.2e8±2e8↑	1.3e8±1e8↑	3e10±7e10↑
f_9	**1.3e7±2e6**	4.5e8±5e7↑	1.7e9±2e8↑	1.0e8±9e6↑	1.0e9±1e9↑
f_{10}	**1.8e3±1e2**	1.1e4±4e2↑	5.2e3±2e3↑	4.1e3±1e3↑	4.3e3±2e3↑
f_{11}	**2.0e1±3e0**	2.6e1±1e0↑	2.0e2±2e0↑	1.0e2±1e2↑	1.9e2±3e1↑
f_{12}	**2.0e1±2e1**	9.9e4±1e4↑	8.7e5±1e5↑	9.1e3±1e3↑	4.8e5±5e5↑
f_{13}	**5.3e2±1e2**	5.3e3±3e3↑	3.2e4±3e4↑	5.4e3±3e3↑	8.6e4±2e5↑
f_{14}	**3.1e7±3e6**	9.8e8±8e7↑	3.6e9±5e8↑	3.0e8±2e7↑	2.2e9±2e9↑
f_{15}	**3.2e3±2e2**	1.2e4±7e2↑	1.2e4±2e3↑	1.3e4±2e2↑	1.1e4±3e3↑
f_{16}	**2.0e1±3e0**	6.9e1±5e0↑	4.0e2±3e0↑	2.0e2±2e2↑	3.6e2±1e2↑
f_{17}	**6.7e1±9e1**	3.1e5±2e4↑	1.8e6±2e5↑	7.5e4±5e3↑	9.7e5±1e6↑
f_{18}	**1.4e3±2e2**	3.5e4±1e4↑	1.1e5±3e4↑	1.4e4±1e4↑	7.8e4±2e5↑
f_{19}	1.3e6±1e5	1.1e6±6e4↓	3.0e6±4e5↑	1.6e6±1e6	2.7e6±3e6↑
f_{20}	2.0e9±2e9	4.5e3±8e2↓	1.8e5±2e5↓	**2.3e3±2e2**↓	5.4e3±1e4↓
R^+	—	176.0	187.0	184.0	188.0
R^-	—	34.0	23.0	26.0	22.0
p-value	—	8.03e-03	2.20e-03	3.19e-03	1.94e-03

CEC'2013 Functions

F	CCFR-IDG2	DECC-G	MLCC	DECC-D	DECC-DML
f_1	2e-5±5e-6	3e-6±2e-6↓	1e-6±6e-7↓	1e-17±1e-17↓	7e-8±3e-7↓
f_2	3.6e2±2e1	1.3e3±3e1↑	2e-2±4e-2↓	7.1e1±3e1↓	4.9e0±2e1↓
f_3	2.1e1±1e-2	2.0e1±7e-3↓	2.0e1±9e-4↓	2.0e1±2e-3↓	2.0e1±2e-2↓
f_4	**9.6e7±4e7**	2e11±1e11↑	2e12±8e11↑	3e10±2e10↑	1e12±1e12↑
f_5	**2.8e6±3e5**	8.6e6±1e6↑	1.9e7±5e6↑	6.1e6±2e6↑	1.9e7±8e6↑
f_6	1.1e6±1e3	1.1e6±1e3↓	1.1e6±3e3↓	1.1e6±2e3↓	**1.0e6±5e3**↓
f_7	**2.0e7±3e7**	1.0e9±5e8↑	8.4e9±4e9↑	9.0e7±4e7↑	3.7e9±5e9↑
f_8	**7e10±1e11**	9e15±4e15↑	8e16±4e16↑	2e14±9e13↑	5e16±8e16↑
f_9	**1.9e8±3e7**	6.1e8±1e8↑	1.2e9±3e8↑	5.1e8±1e8↑	1.2e9±4e8↑
f_{10}	9.5e7±2e5	9.3e7±5e5↓	9.3e7±5e5↓	9.3e7±6e5↓	9.3e7±6e5↓
f_{11}	**4e8±3e8**	2e11±9e10↑	1e12±5e11↑	9e8±5e8↑	6e11±7e11↑
f_{12}	1.6e9±2e9	4.4e3±7e2↓	8.8e4±3e4↓	**2.3e3±2e2**↓	5.2e3±1e4↓
f_{13}	**1.2e9±6e8**	9.6e9±3e9↑	5e10±1e10↑	1.7e9±5e8↑	2e10±2e10↑
f_{14}	3.4e9±3e9	2e11±5e10↑	9e11±4e11↑	7.4e9±9e9	2e11±5e11↑
f_{15}	9.8e6±4e6	1.2e7±1e6↑	3.7e8±3e8↑	**6.9e6±7e5**↓	3e10±1e11↑
R^+	—	98.0	96.0	87.0	97.0
R^-	—	22.0	24.0	33.0	23.0
p-value	—	3.02e-02	4.13e-02	1.35e-01	3.53e-02

The symbols ↑ and ↓ have similar meanings as in Table I.

TABLE V
AVERAGE RANKINGS ON THE CEC'2010 AND CEC'2013 FUNCTIONS (FRIEDMAN TEST). THE BEST RESULT IS IN BOLD FONT.

	CCFR-IDG2	DECC-G	MLCC	DECC-D	DECC-DML	p-value
Average Ranking	**2.1429**	3.0286	4.0000	2.3143	3.5143	5.66e-07

soid function [53] is provided in Sect. II in the supplementary material listed in the appendix. The results show that for CCFR-I, the number of function evaluations increases linearly as the dimensionality of the function and the number of subcomponents increase. CBCC1-I, CBCC2-I and CC-I have similar performance to CCFR-I, but for CCFR-I, as the dimensionality of the function and the number of subcomponents increase, the number of function evaluations increases less rapidly than the other three CC algorithms.

C. CCFR with IDG2

The experimental results of CCFR with two grouping methods (provided in Sect. III in the supplementary material listed in the appendix) show that a high grouping accuracy can improve the performance of CCFR, especially for nonseparable variables.

In this section, the performance of CCFR-IDG2 is presented. IDG2 [54], which is an improved variant of differential grouping (DG) [28], is able to group interdependent variables together with a very high accuracy and correctly identify the indirect interaction between decision variables. CCFR-IDG2 is compared with seven CC algorithms (DECC-G [13], MLCC [49], DECC-D [23], DECC-DML [23], DECC [28], CBCC1 [9] and CBCC2 [9]) and two memetic algorithms (MA-SW-Chains [50] and MOS-CEC2013 [51]). It is shown in [55] that the two memetic algorithms are competitive for solving large-scale optimization problems. Note that, for the algorithms with IDG2, the function evaluations spent on groupings are counted as part of the computational budget.

Table IV summarizes the results of CCFR-IDG2, DECC-G, MLCC, DECC-D and DECC-DML. CCFR-IDG2 performs significantly better than the other peer algorithms by several orders of magnitude on all the CEC'2010 partially separable functions (f_4–f_{18}) and most of the CEC'2013 partially separable functions (f_4–f_{11}). This indicates that an efficient grouping method and an efficient resource allocation strategy can help CCFR achieve competitive performance. The average ranking of CCFR-IDG2 is the best among the five CC algorithms on the CEC'2010 and CEC'2013 functions (see Table V).

CCFR is compared with CBCC1, CBCC2 and DECC, which adopt two grouping methods (i.e., DG [28] and IDG2 [54]). The detailed results are provided in Sect. III in the supplementary material listed in the appendix. For IDG2, the comparison results are similar to the comparison results between CCFR-I and its competitors (CBCC1-I, CBCC2-I and CC-I) in Sect. IV-B. The results show that CCFR-IDG2 performs significantly better than CBCC1-IDG2, CBCC2-IDG2 and DECC-IDG2 on most of the fully separable and partially separable functions. The overall performance of CCFR-DG is also better than CBCC1-DG, CBCC2-DG and DECC-DG on the CEC'2010 and CEC'2013 functions. The algorithms with IDG2 perform better than the ones with DG. This is because IDG2 is able to identify interdependence between variables with higher accuracies.

The comparison between CCFR-IDG2 and the two memetic algorithms (MA-SW-Chains and MOS-CEC2013) is provided in Sect. IV in the supplementary material listed in the appendix. The experimental results show that the overall performance of CCFR-IDG2 is worse than MA-SW-Chains and MOS-CEC2013 on the CEC'2013 functions. However, when we replace SaNSDE with another optimizer (i.e., CMAES [56]), the performance of CCFR-IDG2 is improved. Overall, CCFR-IDG2 with CMAES performs better than MA-SW-Chains and MOS-CEC2013 on both the CEC'2010 and CEC'2013 functions.

V. CONCLUSION

In this paper, we presented a new CC framework named CCFR for tackling large-scale global optimization problems. CCFR aims to make efficient use of computational resources among subpopulations. Unlike the traditional CC where the computational resources are equally allocated among subpopulations and CBCC where the computational resources are allocated according to the accumulated contributions of subpopulations from the beginning of the evolutionary process, CCFR allocates resources to subpopulations according to the previous and current contributions of the subpopulations. The CEC'2010 and CEC'2013 large-scale benchmark functions were used to evaluate the performance of CCFR. From our experimental results, several conclusions can be drawn.

Firstly, CCFR can detect stagnant subpopulations and save computational cost on stagnant subpopulations. Secondly, according to the previous and current contributions of subpopulations to the improvement of the best overall objective value, CCFR can make a more efficient computational resource allocation among subpopulations and obtain better solutions than other CC frameworks. Finally, the performance of CCFR depends on the performance of grouping methods. Grouping the interdependent decision variables together with a high accuracy can improve the performance of CCFR. CCFR with an improved differential grouping method is a highly competitive CC algorithm for solving the large-scale optimization problems.

In the future, we are planning to investigate the potential of using the racing algorithm [57], reinforcement learning [58] and the techniques adopted in adaptive selection of operators [59] for allocating computational resources among subpopulations.

Appendix
Supplementary Material Available on the Web

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

1) The sensitivity study of the parameter U of CCFR.
2) The scalability study of CCFR.
3) The performance of CCFR with DG and IDG2.
4) The comparison between CCFR-IDG2 and non-CC algorithms.

Acknowledgement

The authors would like to thank Dr. Bahar Salehi at Dept. of Computing and Information Systems, University of Melbourne, for carefully proofreading this paper.

References

[1] R. Sarker, M. Mohammadian, and X. Yao, Eds., *Evolutionary Optimization*. Springer US, 2002.

[2] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution," in *IEEE Congress on Evolutionary Computation*, 2001, pp. 1101–1108.

[3] A. Vicini and D. Quagliarella, "Airfoil and wing design through hybrid optimization strategies," *AIAA journal*, vol. 37, no. 5, pp. 634–641, 1999.

[4] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Operations Research*, vol. 8, no. 1, pp. 101–111, 1960.

[5] A. Griewank and P. Toint, "Partitioned variable metric updates for large structured optimization problems," *Numerische Mathematik*, vol. 39, no. 1, pp. 119–137, 1982.

[6] M. Z. Ali, N. H. Awad, and P. N. Suganthan, "Multi-population differential evolution with balanced ensemble of mutation strategies for large-scale global optimization," *Applied Soft Computing*, vol. 33, pp. 304–327, 2015.

[7] M. A. Potter and K. A. D. Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving from Nature*, 1994, pp. 249–257.

[8] P. L. Toint, "Test problems for partially separable optimization and results for the routine PSPMIN," The University of Namur, Department of Mathematics, Belgium, Tech. Rep., 1983.

[9] M. N. Omidvar, X. Li, and X. Yao, "Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms," in *Genetic and Evolutionary Computation Conference*, 2011, pp. 1115–1122.

[10] Y. Chen, T.-L. Yu, K. Sastry, and D. E. Goldberg, "A survey of linkage learning techniques in genetic and evolutionary algorithms," University of Illinois at Urbana-Champaign, Urbana IL, Tech. Rep., 2007.

[11] T. Weise, R. Chiong, and K. Tang, "Evolutionary optimization: Pitfalls and booby traps," *Journal of Computer Science and Technology*, vol. 27, no. 5, pp. 907–936, 2012.

[12] R. Salomon, "Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms," *Biosystems*, vol. 39, no. 3, pp. 263 – 278, 1996.

[13] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences*, vol. 178, no. 15, pp. 2985–2999, 2008.

[14] Y. Mei, M. 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, pp. 13:1–13:24, 2016.

[15] F. Van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[16] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, IEEE International Conference on*, 1995, pp. 1942–1948.

[17] Y. Shi, H. Teng, and Z. Li, "Cooperative co-evolutionary differential evolution for function optimization," in *Advances in natural computation*. Springer, 2005, pp. 1080–1088.

[18] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[19] M. Omidvar, X. Li, Z. Yang, and X. Yao, "Cooperative co-evolution for large scale optimization through more frequent random grouping," in *IEEE Congress on Evolutionary Computation*, 2010, pp. 1–8.

[20] Z. Yang, K. Tang, and X. Yao, "Multilevel cooperative coevolution for large scale optimization," in *IEEE Congress on Evolutionary Computation*, 2008, pp. 1663–1670.

[21] X. Li and X. Yao, "Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms," in *IEEE Congress on Evolutionary Computation*, 2009, pp. 1546–1553.

[22] ——, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 2, pp. 210–224, 2012.

[23] M. N. Omidvar, X. Li, and X. Yao, "Cooperative co-evolution with delta grouping for large scale non-separable function optimization," in *IEEE Congress on Evolutionary Computation*, 2010, pp. 1–8.

[24] 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," Nature Inspired Computation and Applications Laboratory, Tech. Rep., 2010.

[25] K. Weicker and N. Weicker, "On the improvement of coevolutionary optimizers by learning variable interdependencies," in *IEEE Congress on Evolutionary Computation*, 1999, pp. 1627–1632.

[26] W. Chen, T. Weise, Z. Yang, and K. Tang, "Large-scale global optimization using cooperative coevolution with variable interaction learning," in *Parallel Problem Solving from Nature*, 2010, vol. 6239, pp. 300–309.

[27] M. Tezuka, M. Munetomo, and K. Akama, "Linkage identification by nonlinearity check for real-coded genetic algorithms," in *Conference on Genetic and Evolutionary Computation*, 2004, vol. 3103, pp. 222–233.

[28] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 378–393, 2014.

[29] M. N. Omidvar, Y. Mei, and X. Li, "Effective decomposition of large-scale separable continuous functions for cooperative co-evolutionary algorithms," in *IEEE Congress on Evolutionary Computation*, 2014, pp. 1305–1312.

[30] B. Kazimipour, M. N. Omidvar, X. Li, and A. Qin, "A sensitivity analysis of contribution-based cooperative co-evolutionary algorithms," in *IEEE Congress on Evolutionary Computation*, 2015, pp. 1–8.

[31] M. Yang, C. Li, Z. Cai, and J. Guan, "Differential evolution with auto-enhanced population diversity," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 302–315, 2015.

[32] F. Peng, K. Tang, G. Chen, and X. Yao, "Multi-start jade with knowledge transfer for numerical optimization," in *IEEE Congress on Evolutionary Computation*, 2009, pp. 1889–1895.

[33] Y.-l. Li and J. Zhang, "A new differential evolution algorithm with dynamic population partition and local restart," in *Genetic and Evolutionary Computation Conference*, 2011, pp. 1085–1092.

[34] M. Vasile, E. Minisci, and M. Locatelli, "An inflationary differential evolution algorithm for space trajectory optimization," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 267–281, 2011.

[35] M. Zhabitsky and E. Zhabitskaya, "Asynchronous differential evolution with adaptive correlation matrix," in *Genetic and Evolutionary Computation Conference*, 2013, pp. 455–462.

[36] S.-M. Guo, C.-C. Yang, P.-H. Hsu, and J.-H. Tsai, "Improving differential evolution with a successful-parent-selecting framework," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 717–730, 2015.

[37] S.-M. Guo, J.-H. Tsai, C.-C. Yang, and P.-H. Hsu, "A self-optimization approach for l-shade incorporated with eigenvector-based crossover and successful-parent-selecting framework on cec 2015 benchmark set," in *IEEE Congress on Evolutionary Computation*, 2015, pp. 1003–1010.

[38] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[39] M. Yang, Z. Cai, C. Li, and J. Guan, "An improved adaptive differential evolution algorithm with population adaptation," in *Genetic and Evolutionary Computation Conference*, 2013, pp. 145–152.

[40] D. Thierens, "Adaptive strategies for operator allocation," in *Parameter Setting in Evolutionary Algorithms*, ser. Studies in Computational Intelligence, F. Lobo, C. F. Lima, and Z. Michalewicz, Eds. Springer Berlin Heidelberg, 2007, vol. 54, pp. 77–90.

[41] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[42] K. Li, A. Fialho, S. Kwong, and Q. Zhang, "Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 114–130, 2014.

[43] Q. Zhang, W. Liu, and H. Li, "The performance of a new version of moea/d on cec09 unconstrained mop test instances," in *IEEE Congress on Evolutionary Computation*, 2009, pp. 203–208.

[44] A. Zhou and Q. Zhang, "Are all the subproblems equally important? resource allocation in decomposition-based multiobjective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 52–64, 2016.

[45] A. Fialho, M. Schoenauer, and M. Sebag, "Analysis of adaptive operator selection techniques on the royal road and long k-path problems," in *Genetic and Evolutionary Computation Conference*, 2009, pp. 779–786.

[46] F.-M. De Rainville, M. Sebag, C. Gagné, M. Schoenauer, and D. Laurendeau, "Sustainable cooperative coevolution with a multi-armed bandit," in *Genetic and Evolutionary Computation Conference*, 2013, pp. 1517–1524.

[47] 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," Evolutionary Computation and Machine Learning Group, RMIT University, Australia, Tech. Rep., 2013.

[48] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[49] Z. Yang, K. Tang, and X. Yao, "Multilevel cooperative coevolution for large scale optimization," in *IEEE Congress on Evolutionary Computation*, 2008, pp. 1663–1670.

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

[51] A. LaTorre, S. Muelas, and J.-M. Pena, "Large scale global optimization: Experimental results with mos-based hybrid algorithms," in *IEEE Congress on Evolutionary Computation*, 2013, pp. 2742–2749.

[52] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *IEEE Congress on Evolutionary Computation*, 2008, pp. 1110–1116.

[53] R. Ros and N. Hansen, "A simple modification in cma-es achieving linear time and space complexity," in *Parallel Problem Solving from Nature*, 2008, pp. 296–305.

[54] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao, "IDG: A faster and more accurate differential grouping algorithm," University of Birmingham, School of Computer Science, Tech. Rep. CSR-15-04, September 2015. [Online]. Available: ftp://ftp.cs.bham.ac.uk/pub/tech-reports/2015/CSR-15-04.pdf

[55] A. LaTorre, S. Muelas, and J.-M. Peña, "A comprehensive comparison of large scale global optimizers," *Information Sciences*, vol. 316, pp. 517–549, 2015.

[56] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," *Evol. Comput.*, vol. 11, no. 1, pp. 1–18, 2003.

[57] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp, "A racing algorithm for configuring metaheuristics," in *Genetic and Evolutionary Computation Conference*, 2002, pp. 11–18.

[58] J. Schmidhuber, "A general method for multi-agent reinforcement learning in unrestricted environments," in *Evolutionary computation: theroy and applications*, X. Yao, Ed. World Scientific, 1999, pp. 81–123.

[59] P. A. Consoli, Y. Mei, L. L. Minku, and X. Yao, "Dynamic selection of evolutionary operators based on online learning and fitness landscape analysis," *Soft Computing*, vol. 8886, no. 10, pp. 1–26, 2016.

Ming Yang received the B.Sc., M.Sc., and Ph.D. degrees in computer science from China University of Geosciences, Wuhan, China, in 2005, 2008, and 2012, respectively. He carried out a postdoctoral research at the School of Computer Science, University of Birmingham, U.K. from Dec., 2014 to Dec., 2015.

He is currently an Associate Professor with the School of Computer Science, China University of Geosciences, Wuhan, China. He is also a member of the Hubei Key Laboratory of Intelligent Geo-Information Processing, Wuhan, China. His research interests include swarm intelligence, large-scale optimization, multi-objective optimization and their applications.

Mohammad Nabi Omidvar (S'09-M'15) received his bachelor of computer science with first class honors from RMIT University, Melbourne, Australia, in 2010, and his Ph.D. in computer science from RMIT University in 2015. He also holds a bachelor of applied mathematics from RMIT University. From 2008 to 2015, he was a member of Evolutionary Computing and Machine Learning (ECML) group at RMIT University. He received an Australian Postgraduate Award in 2010 and also received the best Computer Science Honours Thesis award from the School of Computer Science and IT, RMIT University.

He is currently a research fellow in evolutionary computation and a member of Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA) at the School of Computer Science, University of Birmingham. His research interests include large-scale global optimization and many-objective optimization.

Changhe Li (M'12) received the B.Sc. and M.Sc. degrees in computer science from China University of Geosciences, Wuhan, China, in 2005 and 2008, respectively, and the Ph.D. degree in computer science from the University of Leicester, U.K. in July 2011.

He has been an Associate Professor at China University of Geosciences (Wuhan) since 2012. He is the vice chair of the Task Force on Evolutionary Computation in Dynamic and Uncertain Environments. His research interests are evolutionary algorithms with machine leaning, swarm intelligence, multi-modal optimization and dynamic optimization.

Xiaodong Li (M'03-SM'07) received his B.Sc. degree from Xidian University, Xi'an, China, and Ph.D. degree in information science from University of Otago, Dunedin, New Zealand, respectively.

He is currently an Associate Professor at the School of Science (Computer Science and Software Engineering), RMIT University, Melbourne, Australia. His research interests include evolutionary computation, neural networks, data analytics, multiobjective optimization, multimodal optimization, and swarm intelligence. He serves as an Associate Editor of the IEEE Transactions on Evolutionary Computation, Swarm Intelligence (Springer), and International Journal of Swarm Intelligence Research. He is the recipient of 2013 ACM SIGEVO Impact Award.

Zhihua Cai received the B.Sc. degree from Wuhan University, Wuhan, China, in 1986, the M.Sc.degree from the Beijing University of Technology, Beijing, China, in 1992, and the Ph.D. degree from the China University of Geosciences, Wuhan, in 2003.

He is currently a Professor with the School of Computer Science, China University of Geosciences. He has published over 100 research papers in journals and international conferences. His current research interests include data mining, machine learning, evolutionary computation, and their applications.

Borhan Kazimipour (S'12) received the B.Sc. degree in Software Engineering and the M.Sc. degree in Artificial Intelligence from Shiraz University, Shiraz, Iran, in 2007 and 2010, respectively. Since 2012, he has been a member of the Evolutionary Computing and Machine Learning Research Group at RMIT University, Melbourne, Australia, where he is currently working towards the Ph.D. degree in the field of evolutionary optimization.

He is currently a Lecturer at the Faculty of Information Technology, Monash University, Melbourne, Australia. His research interests include evolutionary computation, machine learning and big data analytics.

Xin Yao (M'91-SM'96-F'03) received the B.Sc. degree from the University of Science and Technology of China (USTC), Hefei, China, in 1982; the M.Sc. degree from North China Institute of Computing Technology, Beijing, China, in 1985; and the Ph.D. degree from the USTC, Hefei, in 1990. He was an Associate Lecturer and Lecturer with the USTC, from 1985 to 1990; a Postdoctoral Fellow with the Australian National University, Canberra, Australia, and Commonwealth Scientific and Industrial Research Organization, Melbourne, Australia, from 1990 to 1992; and a Lecturer, Senior Lecturer, and Associate Professor with the University of New South Wales, the Australian Defence Force Academy (ADFA), Canberra, from 1992 to 1999. Since April 1999, he has been a Professor (Chair) of Computer Science with the University of Birmingham, U.K., where he is currently the Director of the Centre of Excellence for Research in Computational Intelligence and Applications. He is also a Professor at the Department of Computer Science and Engineering, Southern University of Science and Technology, China.

He is an IEEE CIS Distinguished Lecturer and a former EIC of the IEEE Transactions on Evolutionary Computation. He was a recipient of the 2001 IEEE Donald G. Fink Prize Paper Award, the 2010 IEEE Transactions on Evolutionary Computation Outstanding Paper Award, the 2011 IEEE Transactions on Neural Networks Outstanding Paper Award, and several other best paper awards.