Hindawi
Complexity
Volume 2018, Article ID 9267054, 15 pages
https://doi.org/10.1155/2018/9267054

WILEY | Hindawi

Research Article

Enhancing Cooperative Coevolution with Selective Multiple Populations for Large-Scale Global Optimization

Xingguang Peng and Yapei Wu

School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an, Shaanxi 710072, China

Correspondence should be addressed to Xingguang Peng; pxg@nwpu.edu.cn

Received 4 May 2018; Accepted 20 June 2018; Published 24 July 2018

Academic Editor: Wenbo Wang

The cooperative coevolution (CC) algorithm features a "divide-and-conquer" problem-solving process. This feature has great potential for large-scale global optimization (LSGO) while inducing some inherent problems of CC if a problem is improperly decomposed. In this work, a novel CC named selective multiple population- (SMP-) based CC (CC-SMP) is proposed to enhance the cooperation of subproblems by addressing two challenges: finding informative collaborators whose fitness and diversity are qualified and adapting to the dynamic landscape. In particular, a CMA-ES-based multipopulation procedure is employed to identify local optima which are then shared as potential informative collaborators. A restart-after-stagnation procedure is incorporated to help the child populations adapt to the dynamic landscape. A biobjective selection is also incorporated to select qualified child populations according to the criteria of informative individuals (fitness and diversity). Only selected child populations are active in the next evolutionary cycle while the others are frozen to save computing resource. In the experimental study, the proposed CC-SMP is compared to 7 state-of-the-art CC algorithms on 20 benchmark functions with 1000 dimensionality. Statistical comparison results figure out significant superiority of the CC-SMP. In addition, behavior of the SMP scheme and sensitivity to the cooperation frequency are also analyzed.

1. Introduction

Large-scale global optimization (LSGO) is a kind of optimization problem that includes hundreds or even thousands of decision variables. The huge number of decision variables may induce the curse of dimensionality and an extremely complex interdependency of variables, both of which are challenging for conventional optimization approaches. In addition, the increase of the decision variables may also lead to a change in the problem's properties.

Cooperative coevolution (CC) algorithms have great potential to conduct LSGO in a divide-and-conquer manner. The original problem is decomposed into several subproblems each of which is optimized by a separate coevolutionary population. Since these subproblems are relative small and optimized concurrently, highly efficient problem solving is expected.

To properly use CC, two issues should be carefully considered: cooperation among subproblems and problem decomposition (also called variable grouping). Since each subproblem just represents a segment of the original problem, one subproblem needs to persistently combine its individuals with collaborators provided by the other subproblems. The resultant whole solutions are then evaluated according to the objective function of the original problem. Obviously, improper collaborators may misguide the fitness evaluation and therefore be harmful for global optimization. In fact, early theoretical works have proved that the conventional CC algorithms are likely to converge to a Nash equilibrium rather than a global optimum if the collaborators cannot provide sufficient information for cooperation.

Plenty of work has been conducted on problem decomposition to minimize the interdependency among subproblems [1–6]. If the subproblems are fully independent from each other, the divide-and-conquer feature of CC algorithms could be directly utilized to optimize large-scale problems. However, fully accurate decomposition is usually impractical especially for large-scale problems. Although some pioneer

works [7–10] have studied the way to address the inhabitation problems of CC through collecting informative collaborators, they cannot be directly applied to LSGO.

In our previous work [11], niching-based multimodal optimization (NMMO) was incorporated into the optimization procedure of each subproblem. The optima found by the NMMO procedures are exchanged among subproblems, which enhances the collection of informative collaborators and significantly improves the performance on LSGO. Nevertheless, in [11] we just trigger the NMMO optimizer occasionally because the convergence efficiency of NMMO is relatively low due to the simultaneous search in different regions. Therefore, the optimization is still mainly conducted by another local search optimizer.

According to our further study, we found that it is not necessary to locate and refine all optima of a subproblem. If some regions are properly selected to be searched, the construction of informative collaborators can still be achieved. In addition, since the number of the regions or nichings are reduced the convergence rate of the NMMO procedure is accordingly improved. Thus, a proper selection scheme is expected to be helpful for optimizing subproblems and providing informative collaborators simultaneously.

Bearing this idea in mind, in this work a selective multipopulation scheme is proposed to enhance the cooperation among subproblems. To find informative collaborators, multiple populations are used to search different regions simultaneously. Different from common multipopulation- or niching-based methods that are essentially for multimodal optimization, in our algorithm only some representative ones are selected according to fitness and diversity, and evolved at each iteration. Thus, only one multipopulation-based optimizer is used to not only find an informative collaborator but also conduct global optimization in each subproblem. Besides, the collaborative information is adaptively updated among subproblems. The proposed algorithm is tested on 20 LSGO benchmark functions [12] launched at the 2010 IEEE Congress on Evolutionary Computation (CEC'10) and compared with 7 state-of-the-art CC algorithms. Experimental results validate the effectiveness.

The remainder of this paper is organized as follows. Background knowledge and motivation are presented in Section 2. In Section 3, the detail of the proposed method is described. Behavior analysis and comparison are given in Section 4. In the last section, we summarize our work and draw the conclusions.

2. Background and Motivation

In this section, we briefly introduce the CC algorithm and CMA-ES. Then, the background knowledge and challenges of CC are presented. We figure out that the optimizer of subproblems should not only provide informative collaborators with good diversity and fitness but also adapt to dynamic landscapes.

2.1. Background

2.1.1. CC Algorithm. The pseudocode of the general CC framework is shown in Algorithm 1. At first, the original problem is decomposed into N subproblems with a certain grouping method, then the context vector is initialized. Note that each subproblem is only a part of the original problem. Therefore, we need collaborative information (CI) from other subproblems to evaluate individuals in the current subproblem. Typically, the collaborative information is the current best solution in each subproblem. After initializing the subproblem populations (SP), subproblems will be optimized in a round-robin fashion. The optimizer could be any evolutionary algorithm. The historical best solution of each subproblem is often stored as the collaboration information. The algorithm terminates when the number of fitness evaluations (FEs) exceeds the maximum number. In line 10, the best solution is output as the final result.

```
1  Decompose the original problem into N subproblems;
2  Randomly initialize the context vector CI;
3  Initialize SP(i), i = 1, ... N;
4  while Termination = false do
5      for i = 1 : N do
6          (CI) = Optimizer(SP(i), CI);
7      end
8  end
9  Best = Output(CI);
10 return Best
```

Algorithm 1: CC Algorithm.

2.1.2. CMA-ES. In the multiple population framework used in this work, each population is evolved by an independent local search optimizer. The covariance matrix adaptation evolution strategy (i.e., CMA-ES) is proposed by Hansen and Ostermeier [13]. By sampling and updating a normal distribution adaptively, the CMA-ES shows a powerful local search ability in continuous space. The main formula of CMA-ES is shown as follows:

$$x_i = m + \sigma N_i(0, C), \tag{1}$$

where m is the sampling center, σ is step size, and C is the covariance matrix which determines the shape of the distribution. The parameters m, σ, and C update in different recombination, cumulation, and selection strategies [14].

2.2. Challenges for CC

2.2.1. Dynamic Landscapes. Due to the divide-and-conquer nature of CC, fitness evaluation in a subproblem is related to the other subproblems, in other words, the fitness landscape is dynamic when there exists interdependency among the decision variables in different subproblems. This can be demonstrated by the following example.

Equation (2) is the Ackley function with D dimensions. A $D \times D$ orthogonal matrix $\mathbf{M}$ is used to rotate the coordinate, which makes the function nonseparable. The Ackley function is widely used to build benchmark functions for assessing optimization methods. In this case, let $D = 4$ and the four decision variables are arbitrarily grouped into two

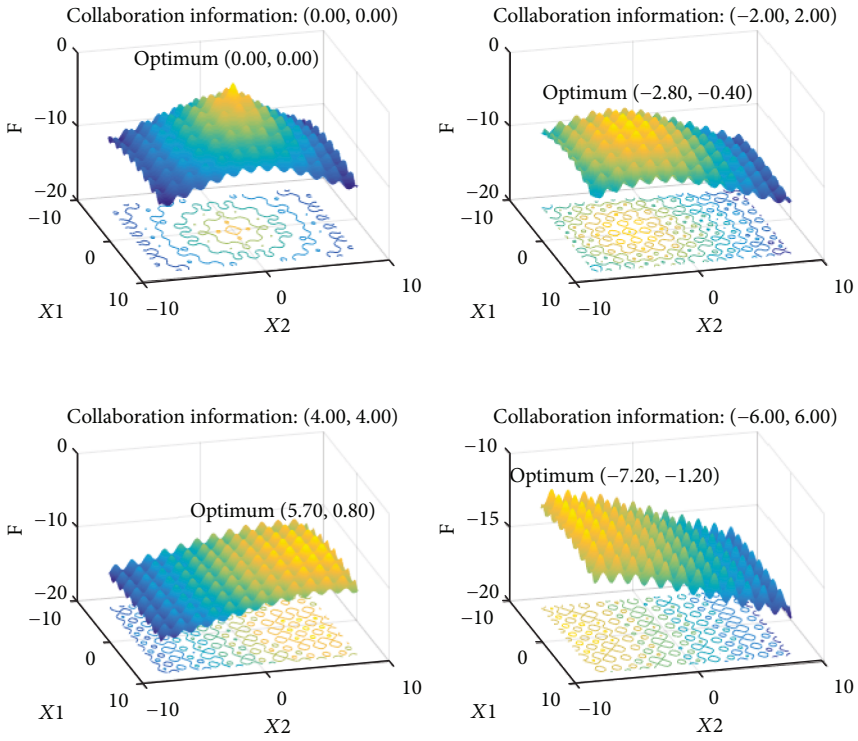

Figure 1: Demonstration of the dynamic nature of CC landscapes.

subproblems (each consists of 2 decision variables). Thus, the resultant two subproblems are interdependent.

$$F_{\text{ackley}}(X) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right), \quad (2)$$

$$F_{\text{rot-ackley}}(X) = F_{\text{ackley}}(Z), Z = X \times \mathbf{M}. \quad (3)$$

Figure 1 shows the corresponding landscapes of the first subproblem when cooperating with four different collaborators from the second subproblem. It can be seen that not only the shape of the landscape but also the location of the optimum may vary accordingly. This reveals the dynamic nature of the optimization procedure in subproblems in the context of CC. Therefore, the optimizer of subproblems should be able to respond to the change of landscapes.

2.2.2. Inherent Problems of CC. The evolutionary game theory (EGT) has been used to theoretically analyze the CC [8, 15]. It has been proved that conventional CC is likely to converge to the Nash equilibrium which is relatively easy to reach. Specifically, if the Nash equilibrium is a local optimum with a large attractive basin, local rather than global optimization will be the resultant of conventional CC. We use the maximum of two quadratics (MTQ) function to visually demonstrate this phenomenon. The MTQ function (formulated in (4)) is a typical function that may lead to

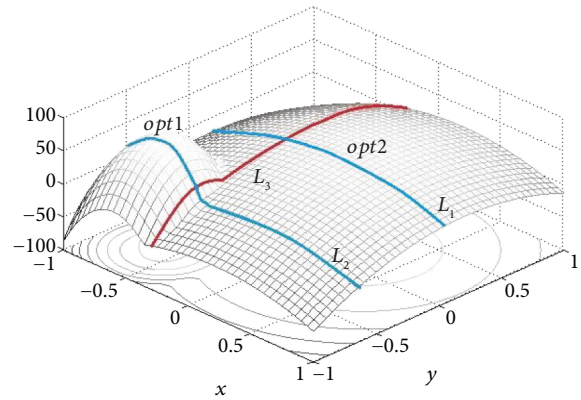

Figure 2: Landscape of the MTQ function with $H_1 = 50$, $S_1 = 16$, $X_1 = 0.2$, $Y_1 = 0.2$, $H_2 = 70$, $S_2 = 2.3$, $X_2 = -0.6$, and $Y_2 = -0.6$.

local optimization of CC and is widely used in the literature [7, 15–18] to test CC algorithms. H determines the height of a peak, S affects the covering area, and $[X, Y]$ is the coordinate of a peak.

$$\text{MTQ}(x, y) = \max \leftarrow \begin{cases} H_1 * \left(1 - \frac{16 * (x - X_1)^2}{S_1} - \frac{16 * (y - Y_1)^2}{S_1}\right), \\ H_1 * \left(1 - \frac{16 * (x - X_2)^2}{S_2} - \frac{16 * (y - Y_2)^2}{S_2}\right), \end{cases} \quad (4)$$

Figure 2 shows the landscape of a MTQ function with two optima. The global optimum (i.e., opt1 : $[-0.6, -0.6]$)

locates on the small peak (covers a small attractive basin) while the local optimum (i.e., opt2 : [0.2,0.2]) locates on the large peak. According to the principle of EGT, it is easy to reach the Nash equilibrium on a large peak (i.e., opt2) where each subproblem could be awarded easily. From the perspective of optimization, global optimization is not necessary until the global optimum locates in a large attractive basin where the Nash equilibrium is easy to reach. Notably, this is not similar to the premature genetic algorithms which could be addressed by maintaining proper population diversity. As for the CC, if the optimization of a subproblem has converged (taking Figure 2 for instance, the population for variable y converges to 0.2), due to the cooperative nature of the CC, the search area for the other subproblem will be particularly restricted (the search space of x is restricted to the solid blue line L_1). In such case, the population cannot search the remaining area where the global optimum may actually be located.

2.3. Motivation. The inherent problem of CC is caused by its cooperative nature. Thus, the proper cooperation scheme should be accordingly designed to provide and utilize informative collaborators. Intuitively, representative information is found and sent to the other subproblem so as to help it "understand" the counterpart. We take Figure 2 as an example, in case that $x = -0.43$ and the search space for y is actually the solid red line L_3. If both the current best optimum (on large peak) and local optimum (on small peak) can be properly obtained and provided to x, the solid blue line L_2 that represents an additional search space of x could be drawn. Along L_2, CC will definitely find the real global optimum. Therefore, the current global and local optima are good representatives of a subproblem. They can be used as informative collaborators to help CC jump out from the locally optimal Nash equilibrium.

Nevertheless, there is no need to find all optima like multimodal optimization. To save the computing resource and improve converging rate, only the optima with great potential to be the real global optimum is worthy of being exploited. Thus, selective property should be additionally incorporated into multimodal optimization.

In this paper, we particularly designed a multipopulation optimizer for CC. In such optimizer, a selection strategy is used to allow potential optima to use more computing resource. Besides, the populations in each subproblem can be adaptively restated to address the dynamic landscape challenge of CC.

3. Proposed Algorithm

In this paper, we propose a selective multipopulation (SMP) scheme to enhance the cooperation among subproblems. The resultant CC algorithm is named CC-SMP for short. The main extension of the CC-SMP is obtaining and sharing more information among subproblems in a uniform selection-based multipopulation framework.

3.1. Algorithm Framework. The pseudocode of the CC-SMP is described in Algorithm 2. First, the original problem is decomposed into several subproblems with a given grouping method (line 1). Then, the coevolutionary populations and information pool are initialized. Specifically, each subproblem is allocated a coevolutionary population with a size of $|group\{i\}|$. These populations are randomly generated according to the grouping result $group\{i\}$ (lines 3–6). Besides, 50 additional complete solutions are randomly generated and evaluated. The fittest solution is copied to the information pool as the initial collaborative information (lines 7–10). The initial coevolutionary populations for subproblems are evaluated via collaborating with initial collaborative information (line 11).

After the initialization procedure, the main optimization procedure is executed (lines 12–24) till the number of fitness evaluations (FEs) exceeds *MaxFes*. A multipopulation-based optimizer is used for each subproblem. This optimizer differs from conventional multipopulation optimizers, since not all of the child populations will have the same opportunity to evolve. A biobjective selection is conducted to identify child populations with good fitness and diversity (lines 14-15) and let them be active to evolve. As for the child populations that are not selected, they will be frozen in the current iteration to save computing resources.

To address the challenge of dynamic landscape, a stagnation check is made for the child populations in each subproblem (line 17). The stagnated child populations are restarted to react in response to the dynamic landscapes. In order to efficiently find and exploit local optima, the CMA-ES, a popular local search algorithm, is used for each child population. Accordingly, several active child populations may simultaneously search different areas of a subproblem. If any child population gets stuck a new local optimum has been found and sufficiently exploited. Since we use the information of optima to construct collaborative information (CI) of CC, a newly raised optimum means the CI should be updated. In such case, CI is updated by selecting elements from the information pool which consists of the historical best solution (*history_best*) and centers of the CMA-ES optimizers (line 20). Those elements are also selected according to fitness and diversity like the selection of active populations to avoid combinational explosion.

3.2. Selective Multiple Populations (SMP). In this work, multiple child populations are used to continually explore and exploit local optima in each subproblem. As mentioned before, not all child populations evolve at every iteration. For improving the converge rate and saving computing resources, only some child populations with great potential will be selected to evolve. The following biobjective selection is proposed to identify potential child populations.

For a given subproblem, N_p child populations are randomly initialized at first with their own evolutionary parameters (e.g., center, step size, and covariance matrix for the CMA-ES). At each generation, a biobjective selection is conducted to select child populations which have competitive fitness and diversity. Then, the selected child populations can evolve while the remaining child populations will be frozen till the next generation. Particularly, the diversity of a given child population is defined as its minimum

Input: Dim, *MaxFes*, *Pop_size*, *Func*, *Ubound*, *Lbound*
Result: One run of CC-SMP
1 $[group, FEused] = grouping(Func)$; $MaxFEs = MaxFEs - FEused$;
2 $FEs = 0$;
3 **for** $i = 1$ *to* $|group|$ **do**
4 $sdim = |group\{i\}|$;
5 $pop\{i\} = Lbound + \text{rand}\,(Pop\_size, sdim) * (Ubound - Lbound)$;
6 **end**
7 $inisolu = Lbound + \text{rand}\,(50, \text{Dim}) * (Ubound - Lbound)$;
8 $inifit = evaluate(inisolu)$;
9 $[\text{min}\_ind] = \text{min}\,(inifit)$;
10 $CI = inisolu(\text{min}\_ind)$;
11 Evaluate all $pop\{i\}$ via collaborating with CI;
12 **while** $FEs < MaxFEs$ **do**
13 **for** $i = 1$ *to* $|group|$ **do**
14 $[active\_ind] = select(pop\{i\}, fitness)$ // see Algorithm 3;
$active\_pop = pop\{i\}(active\_ind)$;
15 **for** $j = 1$ *to* $|active\_pop|$ **do**
16 $[FES1, active\_pop\{j\}, fitness] = \text{Multiple CMA-ES}$
$(group\{i\}, active\_pop\{j\}, CI)$;
17 $[FES2, active\_pop\{j\}, fitness, CI] = check\_restart(active\_pop\{j\}, \text{inf}\_pool)$
// see Algorithm 4;
18 **end**
19 $pop\{i\}(active\_ind) = active\_pop$;
20 $Inf\_pool \leftarrow \{history\_best, centers\_CMA - ES\}$;
21 $FEs = FEs + FES1 + FES2$;
22 **end**
23 **end**

Algorithm 2: Framework of CC-SMP.

Input: $pop, fitness$
Output: *active_ind*
//$[active\_ind] = select(pop, fitness)$
1 **for** $i = 1$ *to* N_p **do**
2 **for** $j = 1$ *to* N_p **do**
3 $D_{ij} = diversity(pop(i), pop(j))$ // (6);
4 **end**
5 **end**
6 $active\_ind = sorting(-D_{ij}, pop, fitness)$ //nondominated sorting;

Algorithm 3: Select Active Populations.

Manhattan distance between other child populations. The Manhattan distance indicates the similarity between vectors. The Manhattan distance between n-dimensional vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ is formulated in (5).

$$d_{ij} = \sum_{k=1}^{n} \left| \mathbf{x}_i(k) - \mathbf{x}_j(k) \right|, \tag{5}$$

where $\mathbf{x}_i(k)$ is the kth dimension of vector $\mathbf{x}_i$. The diversity of $\mathbf{x}_i$ is measured as

$$D(\mathbf{x}_i) = \min_{j=1,\ldots,N_p, j \neq i} d_{ij}, \tag{6}$$

where N_p is the number of vectors. Note that in this work the diversity of a given population refers to its center (in the context of CMA-ES). In other words, the vectors $\mathbf{x}_i$, $i = 1, \ldots, n$, used in (5) are actually the centers of the child populations, and N_p is the number of child populations. Given the diversity $D(\mathbf{x}_i)$ and fitness $f(\mathbf{x}_i)$ of the centers, selection could be conducted according to the following biobjective minimization:

$$\text{argmin}\{-D(\mathbf{x}_i), f(\mathbf{x}_i)\}. \tag{7}$$

The nondominated sorting method used in NSGA-II [19] is employed to rank the centers. Child populations with nondominated centers are selected to evolve in the next iteration. Algorithm 3 shows the detail of the selection process.

Comparing with conventional multipopulation evolutionary algorithms, the SMP is easy to conduct and evolves

Input: $active\_pop\{i\}$, $\inf\_pool$, $fitness$
Output: FEs_2, $active\_pop$, CI, $fitness$
1 $[FEs2, active\_pop\{i\}, CI, fitness] = check\_restart(active\_pop\{i\}, fitness, \inf\_pool)$;
2 $C_1 = subbest(1:20) - subbest(end-19:end)$;
3 $C_2 = submedian(1:20) - submedian(end-19:end)$;
4 $C_3 = \max(xbest) - \min(xbest)$;
5 **if** $C_1 < 0 | C_2 < 0 | C_3 < Tol$ **then**
6 $active\_pop\{i\} = Ubound + Lbound - active\_pop\{i\}$;
7 $[CI] = select(Inf\_pool)$;
8 $[FEs, fitness] = evaluate(active\_pop)$;
9 **end**

Algorithm 4: Check Stagnation and Restart.

efficiently. In conventional algorithms, the management of the child populations including parameters (search space, step size, etc.), life time, and update strategy are relatively complex and additional parameters may also be involved. In contrast, in SMP, selected child populations evolve and restart (as presented in Section 3.3) separately without a merging/splitting operation. The biobjective selection makes the child populations search in the distributed area with good diversity. Besides, selecting potential populations to be active for evolution also saves computing resources and speeds up the rate of convergence.

3.3. Stagnation and Restart. In this work, we propose a restart scheme so that multiple child populations are capable of exploring local optima continually. In addition, this restart scheme enhances the optimizer of each subproblem to react to a dynamic landscape. In particular, the best fitness of active optima is kept to identify stagnation. Stagnated populations are then restarted to maintain the proper diversity which is essential to continually search (exploring new optima and searching in a dynamic landscape).

The pseudocode of stagnation identification is shown in Algorithm 4. As mentioned before, the CMA-ES is used as the optimizer of each child population. Stagnation criteria for a child population are borrowed from [20]. As for a given child population, the best and median fitness values (termed *subbest* and *submedian*, resp.) in the last $(120 + 30D/lamda)$ generations are persistently archived and updated. D is the dimension of the subproblem and *lamda* is the number of the sampling points of CMA-ES in a generation. In this work, *lamda* is set to $[4 + 3 * \ln(D)]$ as suggested in [21].

Three criteria (C_1, C_2, and C_3) are used to check whether a CMA-ES optimizer gets stuck. C_1 and C_2 depend on the data sets in *subbest* and *submedian*. In particular, the 20 newest values in *subbest* and *submedian* should be smaller than the 20 oldest ones. As for C_3, the difference between the maximum and minimum of the recent $(10 + 30D/lamda)$ fittest values should be larger than a predefined threshold *Tol*. If either of these three criteria is not fulfilled, the corresponding CMA-ES optimizer is identified as stagnation.

In this work, we restart a child population in its opposite position [22] (line 6). The opposite position of x is defined as $a + b - x$. $[a, b]$ bounds variable x. Reference [23] shows that hybridization opposition-based learning (OBL) and CC can improve the performance of CC. The key idea of OBL is that the opposite of a candidate solution has relatively high probability to obtain a better fitness value.

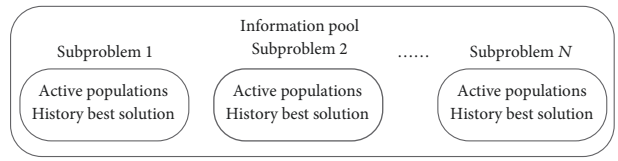

Figure 3: Demonstration of information pool.

3.4. Select Collaborative Information. In the CC algorithm, the cooperation between subproblems is achieved by exchanging collaborative information. Each subproblem is just a part of the original problem, but only complete solutions can be evaluated. To evaluate individuals of a given subproblem, the individuals needs to be combined with collaborative information to form complete solutions. In traditional CC algorithms, only the best individual of each subproblem is provided as collaborative information. As mention before, such cooperation will lead to the convergence to Nash equilibrium which may be suboptimal.

To address this issue, it is necessary to provide collaborators which are more informative to enhance the cooperation among subproblems. According to the description of CC-SMP, each subproblem provides its historical best solution and centers of active CMA-ES populations. An intuitive way to construct complete solutions is to fully mix the collaborators and the individual at hand. However, as the number of subproblems increase, the number of the resultant complete solutions increases exponentially, which may lead to a large number of fitness evaluation. To avoid such combinatorial explosion, the number of informative collaborators should be properly limited. In this work, the collaborative information (historical best solution and centers of active CMA-ES populations) is offered in terms of complete solutions and archived in the information pool. As shown in Figure 3, a complete solution consists of a solution segment of the subproblem and the very collaborator which is used to conduct fitness evaluation. Then, few informative collaborators with good diversity and fitness are selected according to the selection scheme illustrated in Algorithm 3.

Table 1: Description of the CEC'2010 benchmark suite.

Category	Functions	Description	
C1: fully separable 1000D-separable variables	F1	Shifted elliptic function	Unimodal
	F2	Shifted Rastrigin's function	Multimodal
	F3	Shifted Ackley's function	Multimodal
C2: partially-separable Single 50D-nonseparable group 950D-separable variables	F4	Shifted and rotated elliptic function	Unimodal
	F5	Shifted and rotated Rastrigin's function	Multimodal
	F6	Shifted and rotated Ackley's function	Multimodal
	F7	Shifted and Schwefel's problem 1.2	Unimodal
	F8	Shifted and Rosenbrock's function	Multimodal
C3: partially-separable Ten 50D-nonseparable groups 500D-separable variables	F9	Shifted and rotated elliptic function	Unimodal
	F10	Shifted and rotated Rastrigin's function	Multimodal
	F11	Shifted and rotated Ackley's function	Multimodal
	F12	Shifted and Schwefel's problem 1.2	Unimodal
	F13	Shifted and Rosenbrock's function	Multimodal
C4: partially-separable Twenty 50D-nonseparable groups	F14	Shifted and rotated elliptic function	Unimodal
	F15	Shifted and rotated Rastrigin's function	Multimodal
	F16	Shifted and rotated Ackley's function	Multimodal
	F17	Shifted and Schwefel's problem 1.2	Unimodal
	F18	Shifted and Rosenbrock's function	Multimodal
C5: fully-nonseparable 1000D-nonseparable variables	F19	Shifted Schwefel's problem 1.2	Unimodal
	F20	Shifted Rosenbrock's function	Multimodal

4. Experiments

In this section, we test and verify the proposed CC-SMP algorithm. First, the CC-SMP is compared with seven state-of-the-art CC algorithms for LSGO. Second, the behavior of CC-SMP is analyzed in the context of CC. Third, parameter sensitivity is analyzed to further understand the performance of the CC-SMP.

4.1. General Parameter Settings. The following experiments are conducted using the benchmark suite for LSGO which was launched at CEC'10 (2010 IEEE Congress on Evolutionary Computation). This benchmark suite includes twenty 1000-dimensional real-valued functions. As shown in Table 1, different basic functions and separability are considered to generate the benchmark functions. According to the separability, the benchmark functions can be classified into five categories:

(i) C1: *fully separable* (F1–F3)

(ii) C2: *partially separable with a single 50-dimensional nonseparable group* (F4–F8)

(iii) C3: *partially separable with 10 50-dimensional nonseparable groups* (F9–F13)

(iv) C4: *partially separable with 20 50-dimensional nonseparable groups* (F14–F18)

(v) C5: *fully nonseparable* (F19–F20)

Unless otherwise specified, a given algorithm is run for 25 times on a function to obtain the average performance for comparison. As for each run, an algorithm terminates when the number of fitness evaluations (FEs) exceeds 3×10^6. As for the CC-SMP, the subproblems evolve one by one, and each subproblem evolves one generation in a single cycle. The differential grouping (DG) [2] is used to decompose the original problem into several subproblems before the CC-SMP algorithms are conducted.

Note that according to our preliminary experimental results, in each cycle of the proposed CC-SMP the number of active optimizers (i.e., the CMA-ES) is always limited (from 1 to 6) even if there are 50 optimizers. In addition, the number of optimizers will not significantly affect the effectiveness and efficiency, because most of the optimizers will not be selected to be active and run in the optimization procedure according to our selection strategy. To make sure there are enough to be selected, we set such number to be 10 in our following experiments.

4.2. Performance Comparison. Here we compare the CC-SMP with seven state-of-the-art CC algorithms: DECC-D [5], MLCC [24], DECC-DML [5], DECC-DG [2], DECC-I [2], CBCC-1 [25], and CBCC-2 [25]. The grouping methods of these CC algorithms are presented in Table 2. Note that in some algorithms (i.e., CC-SMP, DECC-DG, CBCC-1, and CBCC-2) problem decomposition has been conducted before optimization. In the following experiments, the number of FEs that are used for grouping is not counted. The accurate FEs consumed by variable grouping on each function are given in [2]. The optimizer of these seven large-scale optimization algorithms is the SaNSDE [26]. It is a self-adaptive differential evolution algorithm with neighborhood search.

Table 2: Description of compared algorithms.

Algorithm	Grouping method
DECC-D	Delta grouping
MLCC	Random grouping with a pool of potential subcomponent sizes
DECC-DML	Delta grouping with a pool of potential subcomponent sizes
DECC-DG	Differential grouping (DG)
DECC-I	Ideal grouping
CBCC-1	DECC-DG with contribution-based CC algorithm
CBCC-2	DECC-DG with contribution-based CC algorithm

The mean and standard deviation values of the algorithms over 25 independent runs are compared in Table 3. As for each function, the best result (with a significance level of 0.05) is marked in bold. As can be seen, the proposed CC-SMP performs the best on 12 out of 20 functions followed by DECC-I (4/20), DECC-DG (2/20), and CBCC-2 (2/20). In addition, in Table 4 pairwise Wilcoxon test comparisons between the CC-SMP and each peer algorithm are made on each test function. The CC-SMP performs significantly better in 77.8% (109 out of 140) comparing cases but significantly worse in only 16.4% (23 of 140) of the cases. Therefore, the selective multipopulation strategy effectively enhances the CC algorithm on LSGO problems.

To better understand the dynamics of the CC-SMP, Figure 4 shows the average fitness values of the CC-SMP and three compared algorithms over 25 independent runs. It can also be seen that on most of the test functions the CC-SMP converges more quickly.

The Friedman average rankings are given in Table 5 to conduct multiple statistical comparison according to different separabilities (fully, partially, and nonseparable) and function features (unimodal and multimodal). It can be seen that the CC-SMP also obtains the best ranking results.

4.3. Behavior Analysis. Although ten child populations are initialized for each subproblem, at each iteration, only some of them are selected to be active for evolution. In this section, the behavior of SMP is studied to deeply understand how does the SMP select active populations. For each subproblem, we record the fittest value of each active (selected by SMP) child population during the evolution process. Three typical selective behaviors could be observed in Figures 5–7 which are described as follows:

First, several child populations evolve simultaneously and the best performed one is always selected to be active. Figure 5 shows the dynamics of the selected child populations in the 22nd subproblem when optimizing F1 (forty 25-D subcomponents). It can be seen that the 10th child population is always selected to evolve till the number of FEs reaches about 9×10^5. Actually, the 10th child population has discovered the global optimum and restarted after 9×10^5 FEs (avoiding getting stuck for a long time). This means that the SMP can provide the best-performed child population more evolution opportunity. In addition to the best-performed child population, the diversity-controlling criterion in the SMP can also keep several other child populations from occasionally being active to search the global optimum.

Second, if the landscape is unimodal or the diversity of the child populations is relatively small only, the best-performed child population is selected to persistently evolve. Figure 6 shows the dynamics of the selected child populations in the 17th subproblem when optimizing F9 (unimodal). It can be seen that the 10th child population is always selected to be active. Although several other child populations are also selected at the beginning, only the 10th one is persistently selected after about 1×10^5 FEs. In such case, the corresponding subproblem could be optimized by only one child population and computing resource is saved.

Third, in a multimodal landscape the always-being-selected child populations easily get stuck but can be restarted by the SMP to keep a proper level of diversity. Figure 7 shows the dynamics of the selected child populations in the 29th subproblem when optimizing F10 (multimodal). It can be seen that the stagnation takes place in the 2nd and 7th child population, respectively. However, due to the restart scheme in the SMP both can be selected to be active again. In addition, every time a child population is restarted an alternative one is selected to have the lead role of evolution and to persistently evolve till stagnation.

4.4. Sensitivity Analysis to Cooperation Frequency. According to the description in Section 3.2, the main idea of the SMP is to improve the cooperation among subproblems by sharing collaborators which are more informative. Whenever a subproblem updates its collaborators from the information pool, a cooperation takes place.

In the CC-SMP, a subproblem triggers a cooperation only when it is stagnated and about to be restarted. To further investigate the impact of the frequency with which the subproblems cooperate, in the following experiments the CC-SMP is implemented with four cooperation frequencies, i.e. $M = \{1,100,500, restart\}$. Note that M denotes the number of cycles (line 13–22 in Algorithm 2) that are executed between every two occurrences of cooperation.

Table 6 shows the performance with $M = \{1,100,500, restart\}$. It can be seen that in 13 out of 20 functions the performance changes obviously (the maximal value is larger than 1.5 times the minimal value). In particular, $M = 1$ always leads to a relatively poor performance and therefore an obvious change of performance. This is because $M = 1$ means that the subproblems need to update their collaborators in each cycle. In such case, the landscape may change in a high frequency and there is not sufficient time for the optimizer to explore the new landscape. Accordingly, the quality of the informative collaborators and the resultant cooperation is decreased. This can also be verified by convergence curves on selected functions, as shown in Figure 8. It can be seen that the algorithm with $M = 1$ always obtains the lowest convergence rate.

Although a clear superiority of $M = restart$ (used in CC-SMP by default) is not observed in Table 6 and Figure 8 comparing with $M = \{100,500\}$, the restart strategy implies an adaptive behavior when conducting cooperation. There is

Table 3: Performance comparison on CEC'10 LGSO benchmark functions.

Functions		DECC-D	MLCC	DECC-DML	DECC-DG	DECC-I	CBCC-1	CBCC-2	CC-SMP
F1	Mean	**0.00E+00**	8.24E−07	2.77E−07	2.08E+06	3.83E+05	1.96E+06	6.38E+06	**0.00E+00**
	Std	0.00E+00	4.28E−7	9.60E−07	2.05E+06	6.51E+05	1.99E+09	1.82E+07	0.00E+00
F2	Mean	6.52E+01	**2.61E−03**	1.04E+01	4.22E+03	4.39E+03	4.33E+03	4.18E+03	1.31E+03
	Std	4.47E+01	5.34E−03	2.24E+01	3.80E+02	2.96E+02	3.04E+02	5.38E+02	4.03E+01
F3	Mean	2.29E+00	1.27E−02	2.57E−01	1.09E+01	1.10E+01	1.12E+01	1.10E+01	**0.00E+00**
	Std	1.75E−01	2.65E−02	7.06E−01	8.53E−01	6.23E−01	8.96E−01	7.32E−01	0.00E+00
F4	Mean	2.98E+12	1.17E+14	1.18E+14	5.06E+11	2.71E+10	1.81E+11	**1.65E+10**	1.87E+11
	Std	9.35E+11	4.12E+13	1.69E+14	1.96E+11	1.24E+10	1.08E+11	3.62E+09	5.08E+10
F5	Mean	2.86E+08	5.04E+08	4.99E+08	7.36E+07	6.86E+07	7.02E+07	**6.43E+07**	8.69E+07
	Std	1.08E+08	1.36E+08	1.28E+08	9.56E+06	1.24E+07	1.05E+07	1.31E+07	1.66E+07
F6	Mean	5.89E+06	1.90E+07	1.68E+07	**1.58E+01**	**1.63E+01**	8.14E+04	4.11E+04	1.10E+03
	Std	5.43E+06	2.12E+06	6.08E+06	7.30E−01	9.69E−01	2.84E+05	2.05E+05	2.78E+03
F7	Mean	1.47E+05	4.88E+10	3.42E+10	2.79E+04	**1.17E+04**	1.23E+05	1.26E+10	1.06E+08
	Std	2.47E+05	1.64E+10	5.19E+10	2.03E+04	3.96E+03	1.09E+05	1.48E+10	8.94E+07
F8	Mean	1.27E+08	8.23E+08	3.10E+10	2.78E+07	**8.06E+05**	7.50E+06	3.72E+07	5.83E+07
	Std	1.52E+08	1.92E+08	6.90E+10	3.19E+07	1.63E+06	1.84E+07	3.47E+07	2.65E+07
F9	Mean	1.01E+08	1.69E+09	1.05E+09	3.65E+07	4.76E+07	1.02E+07	3.40E+08	**1.41E+03**
	Std	9.09E+06	2.54E+08	1.13E+09	1.49E+07	5.30E+07	3.84E+06	2.67E+08	1.64E+03
F10	Mean	4.07E+03	5.19E+03	4.30E+03	3.33E+03	3.13E+03	2.59E+03	4.90E+03	**1.59E+03**
	Std	1.26E+03	1.72E+03	1.77E+03	1.92E+02	1.68E+02	1.48E+02	6.37E+02	7.16E+1
F11	Mean	9.98E+01	2.00E+02	1.91E+02	2.64E+01	2.51E+01	2.69E+01	2.75E+01	**0.00E+00**
	Std	1.01E+02	2.24E+00	3.56E+01	2.95E+00	2.72E+00	2.64E+00	3.18E+00	0.00E+00
F12	Mean	9.14E+03	8.68E+05	4.76E+05	3.21E+04	2.44E+04	3.53E+04	5.07E+04	**0.00E+00**
	Std	1.08E+03	1.24E+05	4.69E+05	1.06E+04	7.12E+03	1.11E+04	1.10E+04	0.00E+00
F13	Mean	5.44E+03	3.24E+04	8.62E+04	2.89E+07	1.29E+04	9.06E+04	1.29E+07	**1.51E+03**
	Std	2.76E+03	2.61E+04	1.95E+05	1.57E+07	4.34E+03	6.11E+04	7.36E+06	1.39E+03
F14	Mean	3.00E+08	3.62E+09	2.22E+09	2.10E+07	2.14E+07	2.24E+07	5.35E+09	**0.00E+00**
	Std	2.19E+07	5.43E+08	2.04E+09	2.25E+06	2.06E+06	2.27E+06	6.00E+08	0.00E+00
F15	Mean	1.30E+04	1.17E+04	1.10E+04	2.88E+03	2.84E+03	2.84E+03	3.22E+03	**1.69E+03**
	Std	2.18E+02	2.05E+03	2.77E+03	2.76E+02	1.86E+02	2.65E+02	4.17E+02	3.96E+01
F16	Mean	2.02E+02	3.99E+02	3.62E+02	1.97E+01	1.93E+01	1.87E+01	1.91E+01	**0.00E+00**
	Std	1.58E+02	3.43E+00	1.09E+02	3.61E+00	3.77E+00	3.83E+00	2.76E+00	0.00E+00
F17	Mean	7.47E+04	1.79E+06	9.71E+05	7.76E+00	7.08E+00	1.49E+01	1.24E+02	**0.00E+00**
	Std	4.72E+03	1.78E+05	1.05E+06	1.89E+00	1.76E+00	7.01E+00	5.72E+01	0.00E+00
F18	Mean	1.44E+04	1.07E+05	7.77E+04	2.01E+10	**1.15E+03**	4.10E+09	1.23E+11	2.42E+04
	Std	1.27E+04	2.68E+04	1.75E+05	4.82E+09	1.65E+02	1.83E+09	1.45E+10	1.62E+04
F19	Mean	1.59E+06	2.96E+06	2.70E+06	**9.01E+05**	8.95E+05	9.12E+05	9.11E+05	1.05E+06
	Std	1.32E+06	4.29E+05	3.37E+06	6.14E+04	6.24E+04	7.11E+04	6.02E+04	1.49E+05
F20	Mean	2.27E+03	1.75E+05	5.42E+03	6.53E+08	1.67E+07	1.41E+07	6.97E+09	**1.19E+03**
	Std	2.44E+02	2.08E+05	1.46E+04	6.71E+08	3.30E+07	1.96E+07	1.12E+09	1.76E+02

Table 4: Wilcoxon test with a significant level of 0.05. +, −, and ~ denote significantly better, significantly worse and not significantly different, respectively.

	Function number																			
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
DECC	+	−	+	+	+	+	−	+	+	+	+	+	+	+	+	+	+	−	~	+
MLCC	+	−	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
DECC-DML	+	−	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	~	~	+
DECC-DG	+	+	+	+	−	~	−	−	+	+	+	+	+	+	+	+	+	+	−	+
DECC-I	+	+	+	−	−	~	−	−	+	+	+	+	+	+	+	+	+	−	−	+
CBCC-1	+	+	+	~	−	~	−	−	+	+	+	+	+	+	+	+	+	+	−	+
CBCC-2	+	+	+	−	−	~	+	−	+	+	+	+	+	+	+	+	+	+	−	+

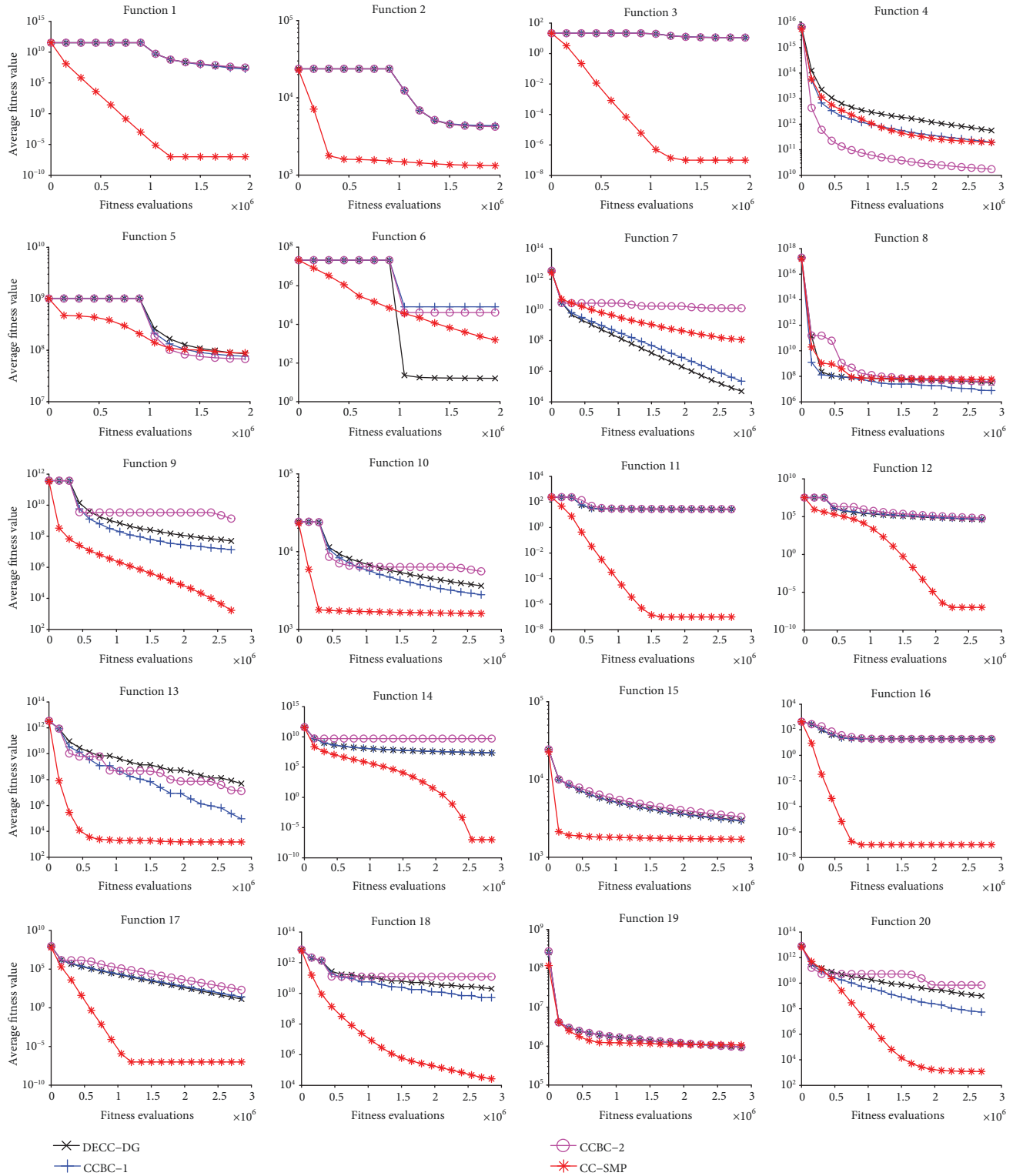

Figure 4: Convergence curves on CEC'10 benchmark function.

no need to carefully set a fixed cooperation frequency. A cooperation is triggered as long as a stagnation takes place. Seen from Figure 9, on different functions the algorithm with a *restart* scheme may adaptively adjust its cooperation frequency. More cooperation takes place when solving multimodal functions (see F5 and F8), because more child populations are needed to search simultaneously to obtain better informative collaborators from the multimodal landscape. These child populations need to jump out from the local optima and update collaborative information, which increases the number of occurrences of cooperation.

Table 5: Friedman average rankings.

Algorithm	Friedman average rankings				
	Fully separable	Partially separable	Nonseparable	Unimodal	Multimodal
DECC-D	2.83	4.13	3.5	3.68	4.0
MLCC	2.33	6.26	5.5	6.37	5.08
DECC-DML	2.66	5.60	4.5	5.62	4.66
DECC-DG	5.66	3.06	3.5	2.75	4.0
CBCC-1	6.33	2.86	4.0	3.0	3.74
CBCC-2	6.0	4.26	4.5	4.5	4.58
CC-SMP	**2.16**	**1.79**	**2.5**	**2.06**	**1.91**

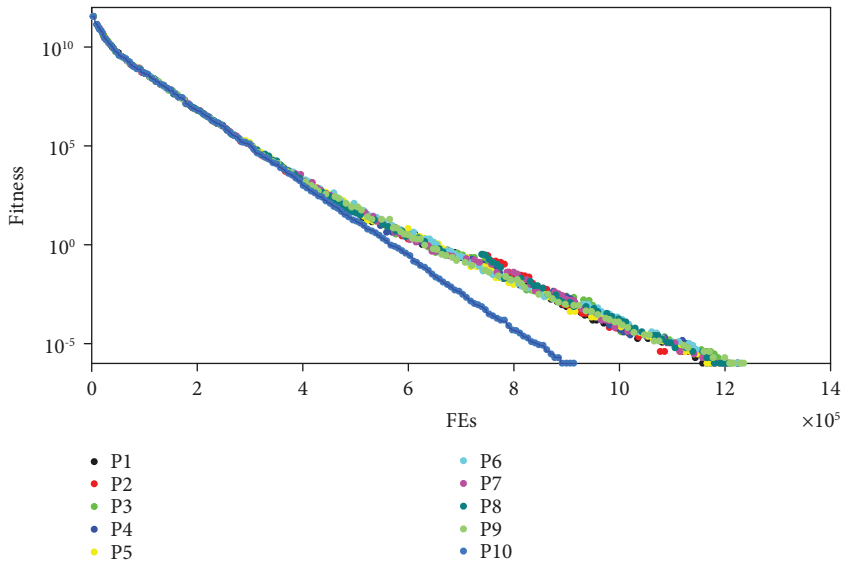

Figure 5: Convergence curves of active child populations in the 22nd subproblem when optimizing F1.

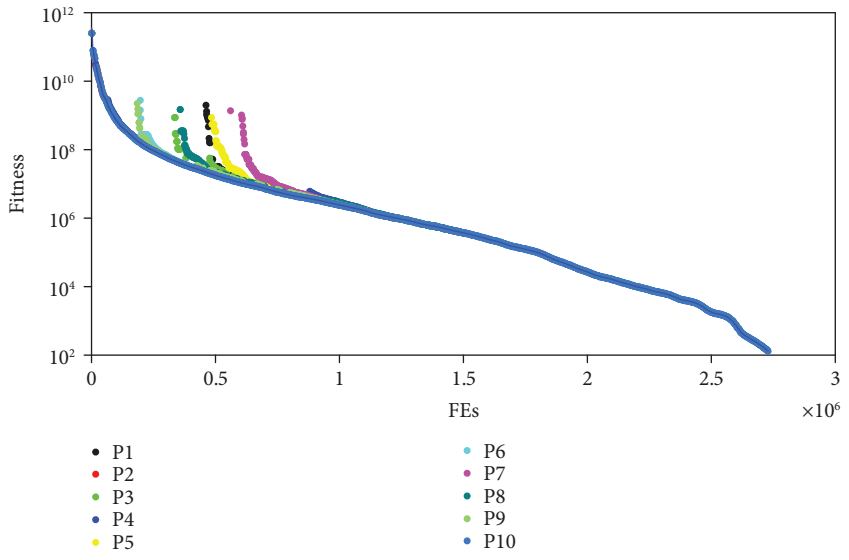

Figure 6: Convergence curves of active child populations in the 17th subproblem when optimizing F9.

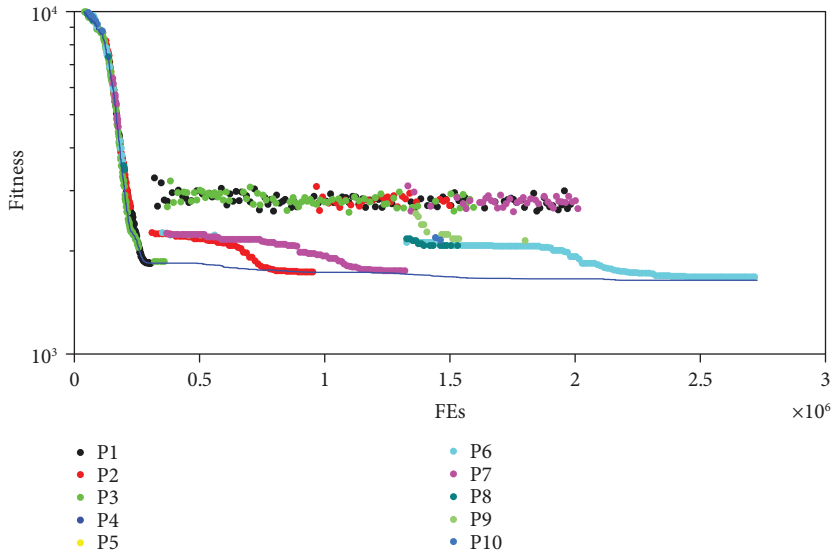

Figure 7: Convergence curves of active child populations in the 29th subproblem when optimizing F10.

Table 6: Performance comparison with different M. Bold values denote the best performance. Bold function number indicates that in such function the maximal value is larger than 1.5 times the minimal value.

Function	1	100	500	Restart
F1	7.1E+04	**0.00E+00**	**0.00E+00**	**0.00E+00**
F2	1.54E+03	1.21E+3	1.36E+3	1.27E+3
F3	3.56E−02	**0.00E+00**	**0.00E+00**	**0.00E+00**
F4	1.35E+11	**6.83E+10**	1.45E+11	2.06E+11
F5	9.47E+07	6.08E+07	6.87E+07	9.17E+07
F6	6.43E+04	3.14E−02	**2.60E−03**	9.94E+01
F7	1.09E+08	4.59E+06	**4.47E+06**	2.41E+07
F8	5.93E+07	4.63E+07	4.71E+07	4.76E+07
F9	1.94E+07	3.34E+02	1.44E+03	**9.31E+01**
F10	1.58E+03	1.43E+03	1.52E+03	1.65E+03
F11	3.60E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**
F12	6.29E+03	**0.00E+00**	**0.00E+00**	**0.00E+00**
F13	1.67E+04	1.29E+03	7.93E+02	**6.88E+02**
F14	1.34E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**
F15	1.62E+03	1.67E+03	1.69E+03	1.77E+03
F16	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F17	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F18	2.26E+06	3.01E+04	2.07E+04	**1.84E+04**
F19	1.06e+06	**5.22E+05**	1.30E+06	9.46E+05
F20	4.12E+08	1.24E+03	**1.02E+03**	1.32E+03

5. Conclusions

In this paper, we have proposed a novel CC named CC-SMP whose cooperation among subproblems is enhanced by the selective multiple population (SMP) scheme. The main motivation of this paper is to provide informative collaborators among subproblems throughout the CC process. By properly utilizing those collaborators, the inherent problem of CC algorithms caused by inaccurate problem decomposition is expected to be addressed. We have argued two challenges to obtain informative collaborators. First, the informative collaborators should be some representative individuals with good fitness and diversity. Second, due to the interactive nature of CC algorithms the landscape of the subproblems is dynamic rather than static.

To address two such challenges, the SMP scheme has been proposed to enhance the cooperation of CC algorithms. A multipopulation scheme with a restart-after-stagnation procedure has been designed to conduct optimization for each subproblem. Several child populations simultaneously search in different local areas and locate corresponding local optima as the informative collaborators. The restart-after-stagnation procedure has been incorporated to help the child populations adapt to a dynamic landscape. To save computing resource and speed up converging rate, not all child populations can evolve all the time. A biobjective selection has also been incorporated to identify the qualified child populations according to the criteria of representative individuals (fitness and diversity). Only selected child populations are active in the next evolutionary cycle while the others are frozen to save computing resource. In the experimental studies, the proposed CC-SMP has been compared with 7 state-of-the-art CC algorithms on twenty 1000D benchmark functions. Both pairwise comparison (Wilcoxon test) and multiple comparison (Friedman ranking) results figure out the significant superiority of the proposed CC-SMP. In addition, behavior analysis of the SMP scheme and sensitivity to the cooperation frequency have been conducted to further understand the CC-SMP.

More work can focus on realizing cooperation among subproblems more efficiently and accurately in the future. Some techniques from the machine learning field may be

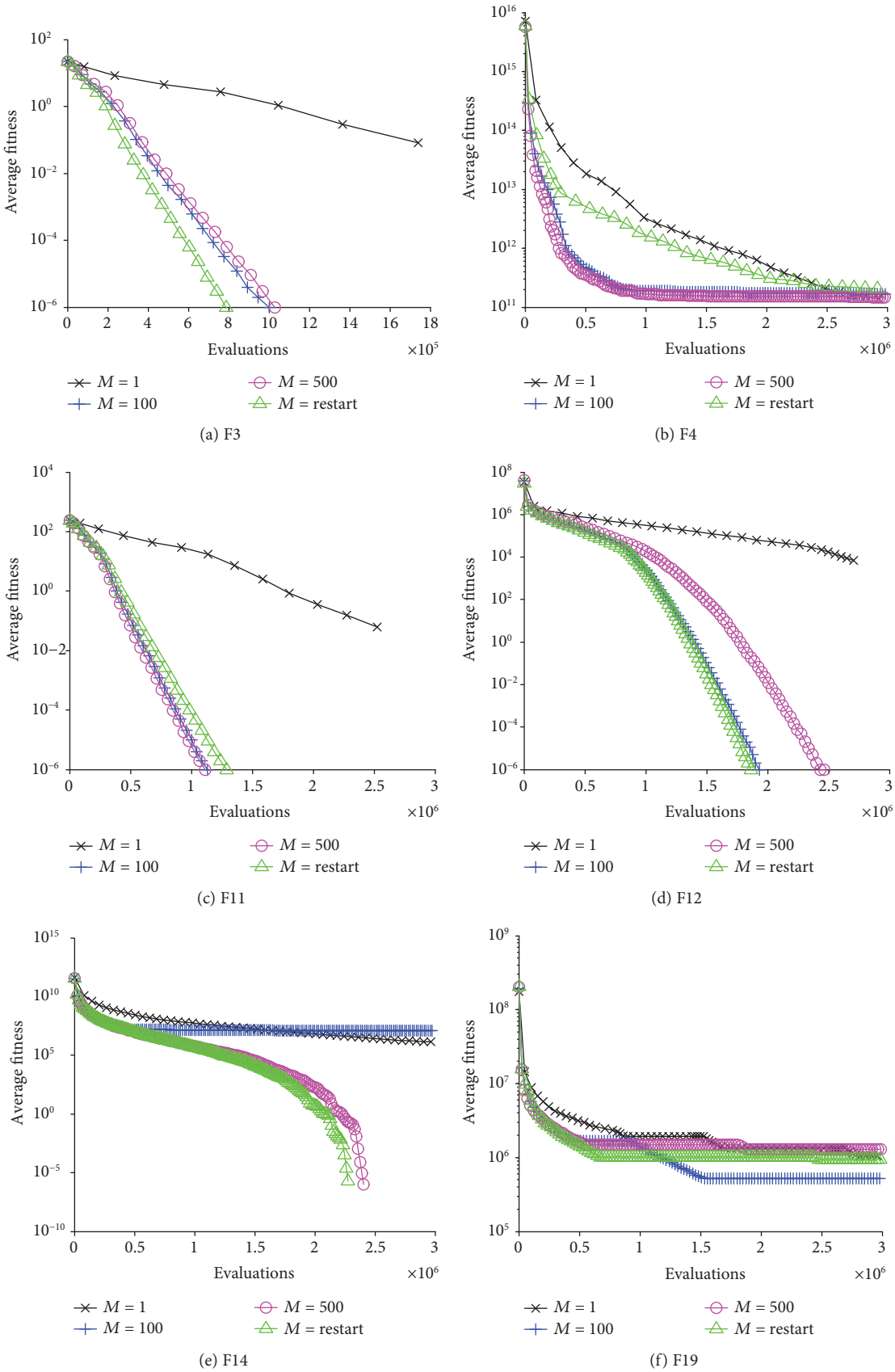

Figure 8: Convergence curves of the CC-SMP with a different collaboration frequency.

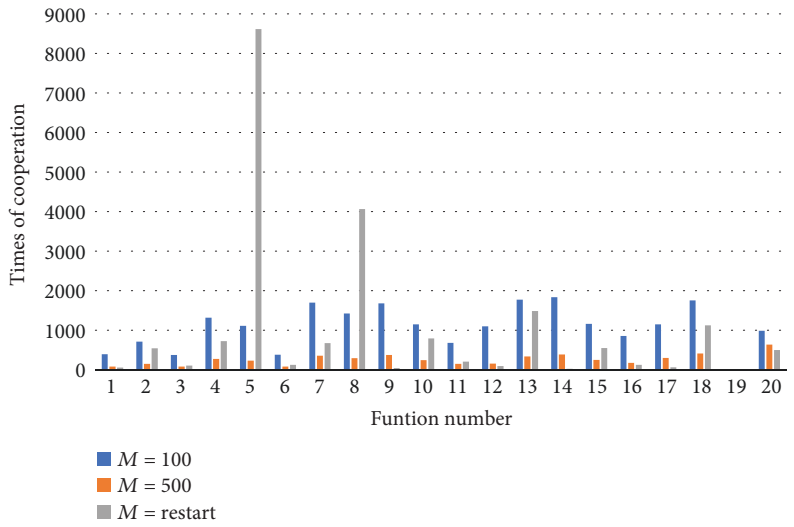

Figure 9: Number of cooperations on 20 test functions.

used to settle the combinatorial explosion problem between subproblems. The data generated in the optimization process may be utilized to improve the quality of collaboration information in our follow-on work.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work has been supported by the National Natural Science Foundation of China (nos. 6117235 and 61473233).

References

[1] 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.

[2] 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.

[3] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *ACM Transactions on Mathematical Software*, vol. 42, no. 2, pp. 1–24, 2016.

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

[5] 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*, pp. 1–8, Barcelona, Spain, 2010, IEEE.

[6] 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 Transactions on Evolutionary Computation*, vol. 21, no. 6, pp. 929–942, 2017.

[7] L. Panait, S. Luke, and R. P. Wiegand, "Biasing coevolutionary search for optimal multiagent behaviors," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 629–645, 2006.

[8] L. Panait, "Theoretical convergence guarantees for cooperative coevolutionary algorithms," *Evolutionary Computation*, vol. 18, no. 4, pp. 581–615, 2010.

[9] E. P. Manning, "Coevolution in a large search space using resource-limited nash memory," in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation - GECCO '10*, pp. 999–1006, Portland, OR, USA, 2010, ACM.

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

[11] X. Peng and Y. Wu, "Large-scale cooperative co-evolution using niching-based multi-modal optimization and adaptive fast clustering," *Swarm and Evolutionary Computation*, vol. 35, pp. 65–77, 2017.

[12] 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," Tech. Rep., Nature Inspired Computation and Applications Laboratory, 2009.

[13] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation," in *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 312–317, Nagoya, Japan, 1996.

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

[15] R. P. Wiegand, *An analysis of cooperative coevolutionary algorithms, [Ph.D. thesis]*, George Mason University, Fairfax, VA, USA, 2003.

[16] A. Bucci and J. B. Pollack, "On identifying global optima in cooperative coevolution," in *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation—GECCO '05*, pp. 539–544, Washington, DC, USA, 2005.

[17] L. Panait and S. Luke, "Selecting informative actions improves cooperative multiagent learning," in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems—AAMAS '06*, pp. 760–766, Hakodate, Japan, 2006.

[18] J. Lewis, E. Hart, and G. Ritchie, "A comparison of dominance mechanisms and simple mutation on non-stationary problems," in *International Conference on Parallel Problem Solving From Nature*, pp. 139–148, Amsterdam, Netherlands, 1998.

[19] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[20] N. Hansen, "Benchmarking a BI-population CMA-ES on the BBOB-2009 noisy testbed," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pp. 2397–2402, Montreal, QC, Canada, 2009.

[21] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *2005 IEEE Congress on Evolutionary Computation*, pp. 1769–1776, Edinburgh, UK, 2005, IEEE.

[22] S. Rahnamayan and G. G. Wang, "Solving large scale optimization problems by opposition-based differential evolution (ODE)," *WSEAS Transactions on Computers*, vol. 7, no. 10, pp. 1792–1804, 2008.

[23] B. Kazimipour, M. N. Omidvar, X. Li, and A. K. Qin, "A novel hybridization of opposition-based learning and cooperative co-evolutionary for large-scale optimization," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2833–2840, Beijing, China, 2014, IEEE.

[24] Z. Yang, K. Tang, and X. Yao, "Multilevel cooperative coevolution for large scale optimization," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 1663–1670, Hong Kong, 2008, IEEE.

[25] M. N. Omidvar, X. Li, and X. Yao, "Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11*, pp. 1115–1122, Dublin, Ireland, 2011.

[26] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 1110–1116, Hong Kong, 2008, IEEE.