Swarm and Evolutionary Computation 35 (2017) 65–77

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Large-scale cooperative co-evolution using niching-based multi-modal optimization and adaptive fast clustering

Xingguang Peng*, Yapei Wu

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

A R T I C L E I N F O

MSC:
00-00-00

Keywords:
Cooperative co-evolutionary algorithm
Large-scale optimization
Information compensation
Clustering

A B S T R A C T

The *divide-and-conquer* problem-solving manner endows the cooperative co-evolutionary (CC) algorithms with a promising perspective for the large-scale global optimization (LSGO). However, by dividing a problem into several sub-components, the co-evolutionary information can be lost to some extent, which may lead to sub-optimization. Thus, information compensation is a crucial aspect of the design of efficient CC algorithms. This paper aims to scale up the information compensation for the LSGO. First, a niching-based multi-modal optimization procedure was introduced into the canonical CC framework to provide more informative collaborators for the sub-components. The information compensation was achieved with these informative collaborators, which is positive for the LSGO. Second, a simple but efficient clustering method was extended to run without manually setting the cut-off distance and identifying clusters. This clustering method, together with a simple scheme, was incorporated to prevent the combinational explosion when mixing the collaborator with a given individual to conduct the fitness evaluation. The effectiveness and superiority of the proposed algorithm were justified by a comprehensive experimental study that compared 8 state-of-the-art large-scale CC algorithms and 8 metaheuristic algorithms on two 1000-dimensional benchmark suites with 20 and 15 test functions, respectively.

1. Introduction

The development of evolutionary algorithms (EAs) has led to the expectation that they can be used to solve more complex optimization problems. The large-scale global optimization (LSGO) is a typical problem found in science and engineering.

The difficulty with the LSGO comes from three main factors. First, the search space of an optimization problem increases exponentially as the dimensionality grows. This may lead to a very large population of EAs and a long search process or even the failure to find the optimum. Second, the topology of the interaction between the decision variables may become extremely complex when the dimensionality is high. Without properly dealing with the interactions between variables, the efficiency of the optimization process may be dramatically affected or even be trapped in the local optima. Third, the increasing number of decision variables may also lead to a change in the problem's properties. Moreover, some general difficulties (such as an expensive objective evaluation, constraints, and noise) may become more difficult to address when a high dimensionality is under consideration.

The cooperative co-evolutionary (CC) algorithm is a branch of the EAs. It conducts optimization in a *divide-and-conquer* manner by decomposing a problem into several relative small sub-components and optimizing them separately. This endows the CC algorithms with a promising perspective for solving the LSGO problems.

Problem decomposition (also known as variable grouping) is essential for making full use of the *divide-and-conquer* for the LSGO. A number of online [1–7] or off-line [8–10] problem decomposition methods have been proposed for large-scale CC algorithms. Another crucial issue is that the optimization procedure drew little attention in the context of large-scale CC algorithms. In fact, it has been theoretically proven that the simple CC algorithm (averaging the fitness values when evaluating an individual by combining the collaborators from the collaborative sub-components) is generally not appropriate for the task of a static, single-objective optimization [11]. This is because the CC algorithm may lose a great deal of landscape information when the interacting variables are divided into separate sub-components. Without the proper landscape information, the co-evolutionary populations will be incorrectly assessed and converge to a Nash equilibrium rather than an optimum according to the evolutionary-game-theory-based mathematical model [11,12].

Therefore, to maintain the global optimization ability of CC algorithms, information should be compensated for during the CC

* Corresponding author.
E-mail addresses: pxg@nwpu.edu.cn (X. Peng), wuyapei@mail.nwpu.edu.cn (Y. Wu).

http://dx.doi.org/10.1016/j.swevo.2017.03.001
Received 1 December 2016; Received in revised form 12 February 2017; Accepted 12 March 2017
Available online 16 March 2017

process. Representative solutions that provide the other sub-components with accurate projections of the joint search space were suggested in [13]. According to this research, the informative collaborator selection scheme [13] and dynamic optimization approach [14] were proposed in previous works to obtain representative solutions. These methods work effectively on 2D benchmark problems under the conventional CC framework, which consists of two co-evolutionary populations. Nevertheless, to compensate the information for the large-scale CC algorithms, two challenges have to be addressed.

The first challenge is to search representative information in complex search space. In the context of the LSGO, the corresponding search space of a sub-component may still be very complex. Therefore, to conduct the information compensation procedure, powerful schemes should be incorporated into the CC framework to effectively obtain the representative information.

The second challenge is to evaluate a co-evolutionary population according to the representative information provided by the other sub-components with reasonable computational resources. When conducting the fitness evaluation in the conventional CC framework, an individual of a sub-component should be completely mixed with the collaborative collaborators provided by the other sub-components. The resultant complete solutions are then evaluated according to the objective function. This may lead to a combinatorial explosion since the number of the complete solutions increases exponentially with the increasing number of sub-components. For example, if a problem is decomposed into 10 sub-components and each of them provides 2 collaborators, $2^{(10-1)} = 85,184$ fitness evaluations (FEs) are required to evaluate a given individual. With such a vast number of FEs, the computational resources will be used up very quickly and the given problem cannot be sufficiently optimized.

This work aims to scale up the information compensation for LSGO problems. Its main contributions are summarized as follows:

1) A modified CC framework that uses niching-based multi-modal optimization (NMMO) to provide informative collaborators to the sub-components: According to the ETG model [11,12], the CC process may stop or become stuck if the co-evolutionary populations reach a Nash equilibrium. However, a sub-optimal Nash equilibrium with large basins of attraction is easier to reach than the global optimal equilibrium. To help CC algorithms converge to the global optimum, the temporary Nash equilibrium is broken by identifying multiple informative collaborators of each sub-component and exchanging them among the sub-components during the CC process. Those multiple informative collaborators are actually the global or local optima of each sub-component identified using an efficient NMMO procedure.

2) A clustering based selection scheme that reduces the number of collaborators used for the fitness evaluation: An adaptive fast clustering algorithm is proposed to cluster a set of informative collaborators collected from the other sub-components. Centroids of the resultant clusters are then used as the most representative collaborators for the fitness evaluation. This scheme not only provides a certain sub-component with sufficient collaborative information, but also saves computational resources.

3) Comprehensive experimental study: The effectiveness of the proposed algorithm has been demonstrated through statistical comparisons with 16 state-of-the-art large-scale EAs on two 1000-dimensional benchmark suites launched for the LSGO competitions in CEC'10 and CEC'13.

The remaining sections of this paper are organized as follows. In Section 2, the genetic framework of the CC algorithms is given. An adaptive fast clustering method is proposed and experimentally demonstrated in Section 3. The details of the proposed algorithm are shown in Section 4 and the experimental study is given in Section 5. Our conclusions and future work are in Section 6.

2. Generic framework of CC algorithms

The generic pseudo-code of CC algorithms is shown in Algorithm 1. Suppose an N-dimensional problem is divided into M sub-components $\mathcal{G}(i)$, i=1,...,M, and each of them is evolved by a co-evolutionary population $\mathbf{CP}(i)$, $i = 1, \ldots, M$. The main difference between the EAs and CC algorithms is that in the CC algorithms, an individual in $\mathbf{CP}(i)$ only codes a segment (according to $\mathcal{G}(i)$) of the corresponding solution. To conduct a co-evolution, each sub-component provides its collaborative information $\mathbf{CI}(i)$, $i = 1, \ldots, M$ with the other sub-components. When evaluating $\mathbf{CP}(i)$, a set of collaborators $\mathbf{CS}(i)$ is first constructed according to the collected $\mathbf{CI}(j)$, $j = 1, \ldots, M$, $j \neq i$ provided by the other sub-components. Then, each individual in $\mathbf{CP}(i)$ is mixed with $\mathbf{CS}(i)$ to construct several complete solutions whose fitness is calculated in terms of the *best-of-N* strategy, i.e., the fitness of the best mixed complete solution.

Algorithm 1. The generic pseudo-code for the CC algorithms.

```
1: Initialize $\mathbf{CP}(i)$, $i = 1, \ldots, M$ for each sub-component.
2: while isTerminated=false do
3:   for $i$=1,...,$M$ do
4:     Construct $\mathbf{CS}(i)$ according to $\mathbf{CI}(j)$, $j \in M$, $j \neq i$;
5:     Evolve $\mathbf{CP}(i)$ with a certain EA by collaborating with $\mathbf{CS}(i)$
       in terms of the best-of-N strategy;
6:     Update $\mathbf{CI}(i)$
7:   end for
8: end while
9: Output results;
```

Notably, in most of the literature about solving LSGO problems with CC algorithms [1–10], each piece of the collaborative information $\mathbf{CI}(i)$ is the current or up-to-date best solution of the corresponding sub-component. This will not lead to the combinatorial explosion problem since every sub-component provides only one collaborator at a time. Actually, the collaboration method is always a challenging issue in regards to CC algorithms. As suggested in a recent theoretical study [12], providing a large enough number of collaborators to the co-evolutionary counterparts is necessary for global convergence, which was empirically demonstrated by identifying the informative collaborators [13]. However, in this situation, the combinational explosion would be induced if the collected collaborators were simply mixed to construct complete solutions. Therefore, providing the sub-components with sufficient informative collaborators while avoiding combinational explosion is a specific challenge for information compensation in the context of large-scale CC algorithms.

3. An adaptive fast clustering method

As shown in Algorithm 1, when conducting the fitness evaluation in the ith sub-component, one has to construct the collaborator set $\mathbf{CS}(i)$ according to $\mathbf{CI}(j)$, $j \in M$, $j \neq i$. This may lead to a combinational explosion (e.g., the complete mixing of $\{\mathbf{CI}(j)\}$), which consumes enormous computational resources (number of FEs). An intuitive way to address this issue is to select highly representative elements from the resultant data set generated from $\{\mathbf{CI}(j)\}$ according to their distributions in the decision space, because the essential purpose of $\{\mathbf{CI}(j)\}$ is to reflect the landscape status of the corresponding sub-components in the context of cooperative co-evolution.

Bearing this idea in mind, we adopt a clustering method to select collaborators in the decision space. Suppose an information pool $\mathbf{POOL}(i)$, $i \in M$ could be obtained for each sub-component with respect to $\mathbf{CI}(j)$, $j \in M$, $j \neq i$ using mixing or any other method. The clustering method attempts to group the elements of $\mathbf{POOL}(i)$ on the bias of their similarity in the decision space. Since the distribution property of those elements is not known beforehand, the clustering method should have the ability to cluster a set of data points into clusters with an arbitrary shape (i.e., the data points are not necessary to meet a Gaussian distribution, as assumed in many clustering methods). In addition, in the context of the LSGO, the cluster method should also perform satisfyingly in a high-dimensional space.

In the K-means and K-medoids methods, a data point is always assigned to the nearest center, which is not suitable for detecting non-spherical clusters. In distribution-based methods, the clustering accuracy greatly depends on the capability of the predefined probability functions to represent the data points. Rodriguez and Laio [15] proposed a fast clustering algorithm by finding the density peaks of the potential clusters (the source code can be downloaded from here[1]). This algorithm is very simple but works effectively and efficiently. For each data point $x(i)$, the algorithm calculates its local density $\rho(i)$ as well as its distance $\delta(i)$ to the closest point with a higher density. The definition of the local density of data point $x(i)$ is shown as below [15]:

$$\rho(i) = \sum_{j} \chi(d_{ij} - d_c), \tag{1}$$

where d_{ij} is the distance between the ith and jth data point, $\chi(\Theta) = 1$ if $\Theta < 0$ and $\chi(\Theta) = 0$ otherwise, and d_c is the cut-off distance. Based on this, $\delta(i)$ is measured by calculating the minimal distance between point i and any other point with a higher density:

$$\delta(i) = \min_{j:\rho(j)>\rho(i)} (d_{ij}). \tag{2}$$

Notably, if a point is the local or global maxima in the density, its δ value is much larger than the typical nearest neighbor distance. Thus, the cluster centers are recognized as the points that have anomalously large δ values. There are two issues that must be addressed to apply this algorithm to different data domains: properly setting the cut-off distance d_c and automatically choosing the centers of the clusters.

For the first issue, the original paper did not provide an efficient way to properly set d_c. In one commentary [16] on the original paper, a potential entropy-based method was proposed to automatically set d_c with respect to a data field. For a data set $x(i)$, $i = 1,\ldots,n$, the potential of ith data point $x(i)$ is calculated as

$$\psi(i) = \Sigma_{j=1}^{n}\left(e^{-\left(\frac{\|x(i)-x(j)\|}{\sigma}\right)^2}\right). \tag{3}$$

Then the entropy H in the data field is

$$H = -\Sigma_{i=1}^{n}\frac{\psi(i)}{z}\log\left(\frac{\psi(i)}{z}\right), \tag{4}$$

where $z = \Sigma_{i=n}^{n}\psi(i)$ is a normalization factor.

Now, we can automatically set d_c by first sampling a set of σ values between the $\min\{\| x(i) - x(j)\|\}$ and $\max\{\| x(i) - x(j)\|\}$, $i, j = 1,\ldots,n$. Then H is calculated according to the different sampling values with Eq. (4) and the one with a minimal H value is selected, termed σ'. Finally, the cut-off value d_c equals $\frac{3}{\sqrt{2}}\sigma'$ according to the suggestion in [17].

For the second issue, as analyzed and suggested in the original paper, $\gamma = \rho \times \delta$ of a set of data points with a uniform distribution should be distributed according to a power law, with an exponent that depends on the dimensionality of the space. In contrast, the distributions of γ for data sets with genuine clusters are strikingly different from those distributed by power laws, especially in the region of a high γ. To select the cluster centers, the original paper provided a decision graph in which the γ values were plotted in descending order. The users had to manually identify the centers according to the anomalously high γ values.

To automatically identify the centers, in this work, we propose a simple procedure as shown in Algorithm 2. Given the ρ and δ values of the data points, γ values can be calculated. Afterwards, the γ values is ranked in descent order (i.e. the *descentSort*() operator), so we get γ' and the corresponding list of indexes **idx**. Now the goal turns to determine how many sequential data points should be selected from the beginning of the ranked data set. To this end, we firstly calculate differences (i.e. the *diff*() operator) between every two adjacent elements and we get $\mathbf{dev} = \{dev(i)\}$, $i = 1,\ldots,n$. Then from the first data point in the ranked data set, extreme outliers check is conducted by comparing $|dev(i) - \overline{\mathbf{dev}}|$ and $\eta \times \mathrm{std}(dev)$. Note that "std" means the calculation of standard deviation. η is a scalar factor and set to 3 in this work. If the former is no longer larger than the later for the ith data point, the first $numCluster = i - 1$ data points is identified as the extreme outliers. This means that the first *numberCluster* elements in **idx** is the indexes of the cluster centers $\mathbf{I}_c$.

Algorithm 2. Cluster centers selection.

```
Require: $\rho(i)$, $\delta(i)$, $i = 1,\ldots,n$
 1: Calculate $\gamma(i) = \rho(i) \times \delta(i)$ for each data points;
 2: [$\gamma'$, **idx**]=*descentSort* ($\{\gamma(i)\}$), $i = 1,\ldots,n$;
 3: *dev*($i$)=*diff* ($\gamma'(i)$), $i = 1,\ldots,n$;
 4: **for** $i$=1,...,$n$ **do**
 5:    **if** $|dev(i) - \overline{\mathbf{dev}}| > \eta \times \mathrm{std}(\mathbf{dev})$ **then**
 6:       $numCluster = i - 1$;
 7:       go to line 10;
 8:    **end if**
 9: **end for**
10: $\mathbf{I_c} = \mathbf{idx}(1{:}\ numCluster)$;
11: Output the indexes of cluster centers, i.e. $\mathbf{I_c}$;
```

Six benchmark data sets are used to illustrate the effectiveness and to verify the adaptive fast clustering (AFC) algorithm. Those data sets can be obtained from a repository site[2] that provides numerous cluster data sets collected from the corresponding papers. 'Flame,' 'R15,' and 'D31′ are 2D shape-cluster data sets with 2, 15, and 31 clusters, respectively. 'Dim32′ and 'Dim1024′ are high dimensional (32D and 1024D) data sets with 16 clusters. 'S3′ is a 2D data set with 16 overlapping clusters. As seen in Fig. 1, the proposed AFC algorithm can correctly cluster all tested data sets.

4. Proposed large-scale CC algorithm

In this section, we present the details of the CC algorithm with niching-based multi-modal optimization and adaptive fast clustering (NMMO-AFC-CC). The algorithm framework is first introduced and then the three main procedures involved in the implementation of the framework are given in details.

4.1. Algorithm framework

In general, given a certain problem decomposition, the NMMO-AFC-CC aims to provide more informative collaborators to the sub-components while avoiding a combinational explosion in the optimization stage. The pseudo-code of the NMMO-AFC-CC is shown in Algorithm 3. For a certain sub-component, there is a main optimizer and a niching-based multi-modal optimization (NMMO) optimizer. The main optimizer is in charge of the co-evolution (details are given in Section 4.2.2). Every main optimizer evolves one iteration in each cycle in a round-robin fashion. If the main optimizing procedure is stuck for a certain number of cycles, the NMMO procedure (details are given in Section 4.2.1) is triggered to temporarily boost the information compensation. In such a situation, the worst individual in the NMMO population is first identified according to the objective func-

[1] http://science.sciencemag.org/content/344/6191/1492

[2] http://cs.joensuu.fi/sipu/datasets/

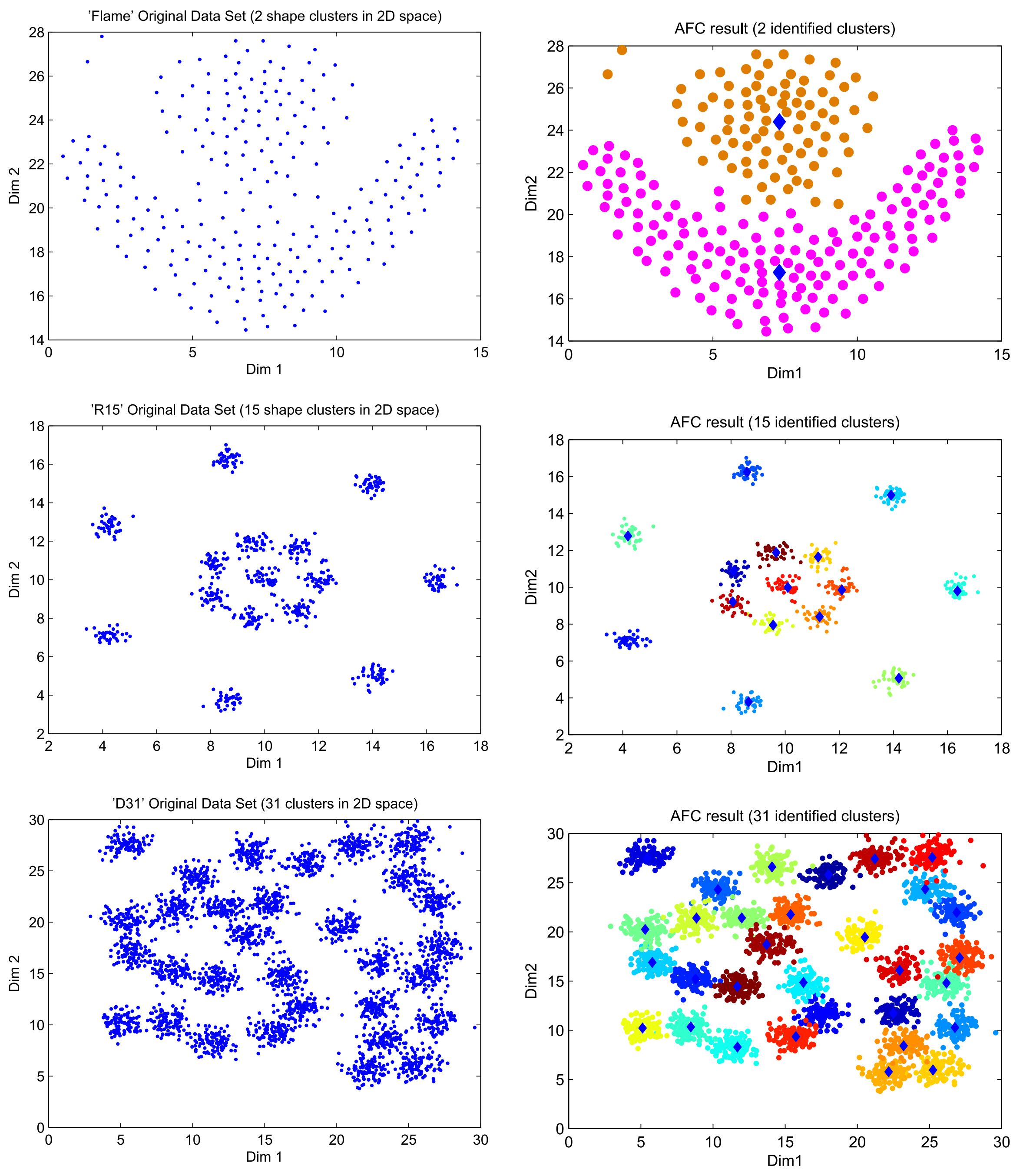

Fig. 1. AFC results on the 6 benchmark data sets.

tion. Afterwards, it is replaced by the up-to-date best solution $B(i)$ of the corresponding sub-component. Then a NMMO optimizer evolves its population, i.e., $\mathbf{S}_{nmmo}(i).\,Pop$, for NG_{nmmo} generations and a set of current optima (local or global), i.e., $\mathbf{S}_{nmmo}(i).\,\mathbf{OPT}$, is obtained. After the NMMO procedure, the main optimizer is heuristically re-started according to the fittest element in $\mathbf{S}_{nmmo}(i).\,\mathbf{OPT}$ and is ready to search from scratch in the next cycle. In the main optimizing procedure, a set of collaborators $\mathbf{Collab}(i)$ is first constructed for every sub-component according to $\{\mathbf{S}_{nmmo}(i).\,\mathbf{OPT}\}$, $i \in M$ and $B(i)$ (details are given in Section 4.2.3). Since the representative information of the collaborative sub-components are used construct the collaborators $\mathbf{Collab}(i)$, the information compensation can be achieved. This is helpful for enhancing the global optimization capability of the resultant large-scale CC algorithm.

Algorithm 3. Algorithm framework of the NMMO-AFC-CC.

```
Require: {$\mathcal{G}$, $MaxFEs$}
  1: /* initialization */
  2: Initiate status of NMMO ($\mathbf{S}_{nmmo}(i)$) and main optimizers ($\mathbf{S}_{main}(i)$);
  3: /* main loop */
  4: while $nFEs \le MaxFEs$ do
  5:    if Main optimizing procedure is stuck then
  6:      /* compensate information */
  7:      for $i$th sub-component, $i$=1,...,$M$ do
  8:         Replace the worst individual in $\mathbf{S}_{nmmo}(i).\,Pop$ with the up-to-date best solution $B(i)$ for the $i$th sub-component;
  9:         NMMO for $i$th sub-component; (Section 4.2.1)
 10:         Update $B(i)$ according to $\mathbf{S}_{nmmo}(i).\,Pop$;
 11:         Heuristic re-start of the $i$th main optimizer;
 12:      end for
```

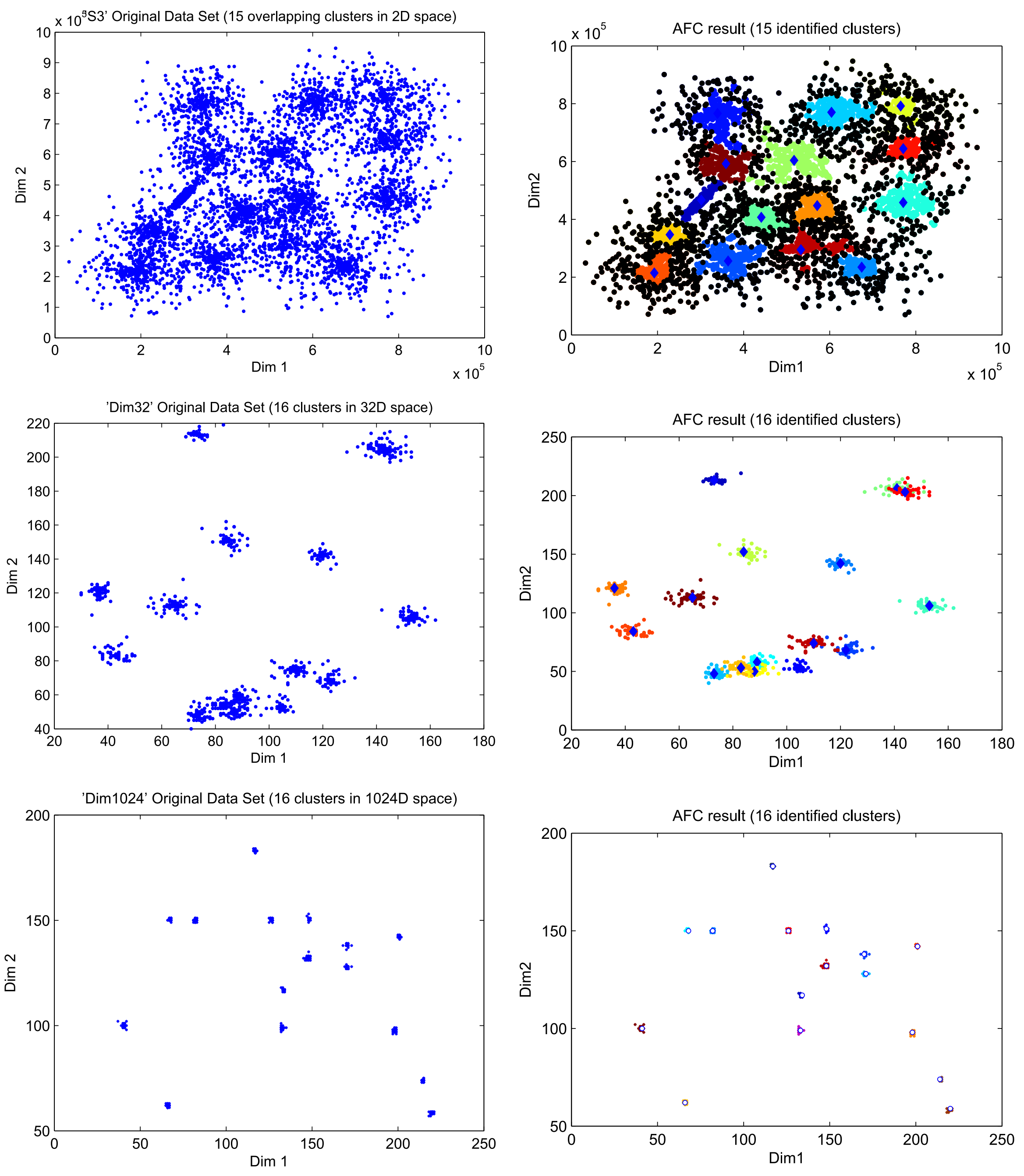

Fig. 1. AFC results on the 6 benchmark data sets.

```
13: else
14:     /* main optimizing process */
15:     for ith sub-component, i=1,...,M do
16:         Construct collaborators Collab(i); (Section 4.2.3)
17:         Execute the ith main optimizer; (Section 4.2.2)
18:         Update B(i) according to S_main(i). Pop;
19:     end for
20:  end if
21: end while
22: Output;
```

4.2. Implementation

In order to implement the NMMO-AFC-CC shown in Algorithm 3, in the following context, we give the details about niching based multi-modal optimizer, main optimizer with heuristic restart and collaborators construction using AFC, respectively.

4.2.1. Niching-based multi-modal optimizer

Niching techniques are popular for multi-modal optimization which require a problem-solving method to maintain the diversity and locate multiple globally optimal solutions. Niching-based EAs promote and maintain the formation of multiple sub-populations within a single population. Classic niching techniques include crowding [18–20], fitness sharing [21], clustering [22,23], restricted tournament selection [24,25], clearing [26], speciation [27] and Euclidean neighborhoods [28,29]. More recently, niching methods have been developed for the Particle Swarm Optimization (PSO) [29,30], Differential Evolution (DE) [31,32], Evolution Strategies (ES) [33], and Estimation of Evolutionary Algorithm (EDA) [34]. A comprehensive review of evolutionary multi-modal optimizations can be seen in [35].

In this paper, a niching migratory multi-swarm optimizer (NMMSO) [36] is introduced to engage with the multi-modal optimization. The NMMSO is designed based on the PSO to meet the following features of an effective multi-modal optimizer [37]: dynamic model maintenance, self-adaptive with few meta-parameters, and a strong local search. It won first place in the CEC' 15 competition on "Niching Methods for Multimodal Optimization".

The high-level pseudo-code for the NMMSO is shown in Fig. 4. The algorithm starts with a single swarm with only one randomly-initialized particle. The main loop consists of five steps. First, the swarms with changed *gbests* and those resulting from the merge are marked. Second, each marked swarm is compared to its nearest neighbor according to the distance between their *gbests* locations. If the neighbor is within *tol* (a preset controlling distance), both of the swarms are directly merged. If not, line-sampling mid-points[3] are used to further detect peaks or valleys and merging is conducted accordingly. Third, a new particle is randomly generated in each swarm with regards to the corresponding *gbest* and the distance to the nearest neighbor swarm. If any swarm is at its capacity, a randomly selected particle is evolved according to the update equations of the PSO. If the number of swarms is larger than *max_inc* (the maximal number of swarms that are allowed to be incremented) *max_inc* swarms either with the best *gbests* or are randomly selected to be incremented. Fourth, swarms at capacity are checked if they should remove a randomly selected particle for the purpose of seeding a new swarm according to the distance between *gbest* and the mid-point. If so, such a particle is copied to seed a new swarm and replaced by the mid-point. If not, *gbest* is compared to the mid-point to see if it should be replaced by the mid-point. In the final step of the main loop, a new swarm must be seeded by either uniform sampling or the uniform crossover of two randomly selected swarm *gbests*. More details about the NMMSO can be found in the original paper [36].

Algorithm 4. High-level pseudo-code of the NMMSO.

```
1: Initiate a swarm with a single particle;
2: while Not terminated do
3:    Attempt to mark each swarm;
4:    Attempt to merge the marked swarms by comparing them to
      their nearest neighbors;
5:    Increment the swarms;
6:    Check the swarms that are at capacity and attempt to seed
      new swarms;
7: Generate a new swarm;
8: end while
9: Output;
```

The implementation of the NMMSO in our NMMO-AFC-CC is directly derived from the original source code provided by the authors of the NMMSO. There are two slight modifications. First, each sub-component maintains an independent archive of the status $\mathbf{S}_{nmmo}$, which temporarily stores the information of the particles and swarms for the NMMSO. Thus, every sub-component can be served in a round-robin fashion with the same NMMSO procedure. Second, the best solution of the main optimizer for the corresponding sub-component in the previous cycle is compared to the nearest *gbest*. If the best solution of the main optimizer is better, the *gbest* is replaced to obtain a further promotion.

4.2.2. Main optimizer with a heuristic re-start

In this paper, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [38] is incorporated into the NMMO-AFC-CC as the main optimizer. The CMA-ES is a powerful optimizer that was extended to conduct the LSGO [7]. It evolves by sampling and updating an embedded covariance matrix. The benefit of the covariance matrix adaptation of the CMA-ES is that it shows a competitive performance for difficult non-linear non-convex continuous optimization problems. We make the following slight modifications to the original CMA-ES to incorporate the CMA-ES into the proposed NMMO-AFC-CC.

[3] Generated in the middle of every 2 *gbests* in the decision space.

Table 1
Description of the compared large-scale CC algorithms.

Algorithm	Grouping method	Optimizer
DECC-D [6]	Delta grouping	SaNSDE
MLCC [3]	Random grouping with a pool of potential subcomponent sizes	SaNSDE
DECC-DML [6]	Delta grouping with a pool of potential subcomponent sizes	SaNSDE [43]
DECC-DG [9]	Differential gourping (DG)	SaNSDE
DECC-I [9]	Ideal gourping	SaNSDE
CBCC−1 CBCC−2	DECC-DG with contribution based CC algorithm [44]	SaNSDE
CC-GDG-CMAES [10]	Global differential gourping extended version of DG	Modified CMA-ES

First, like the NMMSO, each sub-component also maintains a status archive (i.e. $\mathbf{S}_{main}(i)$, $i = 1, \ldots, M$) to store temporary information (information for the covariance matrix adaptation) for the CMA-ES. Thus, every sub-component can be evolved for a single iteration every time in a round-robin fashion.

Second, with the existence of an incorrect problem decomposition and multiple local optima, which may lead to a sub-optimal convergence, the stuck main optimizing procedure can be identified to trigger the NMMO procedure. Afterwards, the informative collaborators can be updated with the NMMO procedure and the stuck co-evolution may become active again. In the context of cooperative co-evolution, if any sub-component (especially the one that contributes to the overall best solution) became stuck, the whole cooperative co-evolution process may be encumbered. Therefore, in this paper, we count the number of cycles for which the overall best solution has not changed. In other words, if the sub-component that contributes to the current overall best solution in a certain cycle cannot obtain a better solution using its corresponding CMA-ES optimizer, such a cycle is counted. The NMMO procedure is triggered if the number of stuck cycles exceeds a certain level L_{stuck}, which can be calculated with regards to the dimensionality $|\mathcal{G}(i)|$ of each sub-component:

$$L_{stuck} = mean\left(30 \times \left\lceil \frac{|\mathbf{S}_{main}(i).\ Pop|}{|\mathcal{G}(i)|} \right\rceil\right),\ i \in M, \tag{5}$$

where the population size, i.e., $|\mathbf{S}_{main}(i).\ Pop|$ equals $4 + \lfloor 3\log(|\mathcal{G}(i)|) \rfloor$ as suggested in [39].

Third, if a promising solution was obtained in the previous NMMO procedure, it should be further improved by an exploitive local searcher. To this end, the corresponding CMA-ES optimizer can be heuristically re-started. As seen in Algorithm 5, if the fittest element in $\mathbf{S}_{nmmo}(i).\ \mathbf{OPT}$ obtained by the corresponding NMMO is better than the best-ever solution (the up-to-date best one since re-initialization) of the corresponding CMA-ES optimizer, the population of CMA-ES (i.e. $\mathbf{S}_{main}(i).\ Pop$) is re-initialized accordingly.

Algorithm 5. Heuristically re-start *i*th CMA-ES.

```
1: if $best\{\mathbf{S}_{nmmo}(i).\ \mathbf{OPT}\}$ $better\_than$ $\mathbf{S}_{main}(i).\ bestever$ then
2:    $\boldsymbol{x}^{(0)} = best\{\mathbf{S}_{nmmo}(i).\ \mathbf{OPT}\}$;
3:    $\boldsymbol{\sigma}^{(0)} = \min(0.3(\mathbf{ub} - \mathbf{lb}), \min(\boldsymbol{x}^{(0)} - lb, ub - \boldsymbol{x}^{(0)}))$; // ub and lb
      are the upper and lower bounds, respectively.
4:    $\mathbf{S}_{main}(i).\ Pop = \boldsymbol{x}^{(0)} + \boldsymbol{\sigma}^{(0)} \bigcirc \mathcal{N}(0, \mathbf{I}_{n\times n})$; // $\bigcirc$is the entrywise
      production, $n = |\mathcal{G}(i)|$.
5: end if
```

Table 2
Performance comparisons on CEC'10 LSGO benchmark functions. The p-values are obtained using tailed Wilcoxon signed ranks test with a significant level of 0.05 for the alternative hypothesis that NMMO-AFC-CC performs better than a compared algorithm. A highlighted *p-value* means the NMMO-AFC-CC's performance is significantly better.

Functions		DECC-D	MLCC	DECC-DML	DECC-DG	DECC-I	CBCC-1	CBCC-2	CC-GDG-CMAES	NMMO-AFC-CC
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	0.00E+00
	Std	0.00E+00	4.28E−07	9.60E−07	2.05E+06	6.51E+05	1.99E+06	1.82E+07	0.00E+00	0.00E+00
	p-value	1.00E+00	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.53E−06**	**6.54E−06**	1.00E+00	/
F2	Mean	6.52E+01	2.61E−03	1.04E+01	4.22E+03	4.39E+03	4.33E+03	4.18E+03	1.60E+03	4.19E+03
	Std	4.47E+01	5.34E−03	2.24E+01	3.80E+02	2.96E+02	3.04E+02	5.38E+02	6.82E+01	1.28E+03
	p-value	1.00E+00	1.00E+00	1.00E+00	5.64E−01	3.73E−01	5.32E−01	5.96E−01	1.00E+00	/
F3	Mean	2.29E+00	1.27E−02	2.57E−01	1.09E+01	1.10E+01	1.12E+01	1.10E+01	2.15E+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	1.48E−02	0.00E+00
	p-value	**6.54E−06**	**6.54E−06**	**6.51E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	/
F4	Mean	2.98E+12	1.17E+14	1.18E+14	5.06E+11	2.71E+10	1.81E+11	1.65E+10	1.52E+10	1.77E+06
	Std	9.35E+11	4.12E+13	1.69E+14	1.96E+11	1.24E+10	1.08E+11	3.62E+09	1.43E+10	1.05E+06
	p-value	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	/
F5	Mean	2.86E+08	5.04E+08	4.99E+08	7.36E+07	6.86E+07	7.02E+07	6.43E+07	9.59E+07	1.56E+08
	Std	1.08E+08	1.36E+08	1.28E+08	9.56E+06	1.24E+07	1.05E+07	1.31E+07	2.23E+07	8.74E+07
	p-value	**1.73E−04**	**6.54E−06**	**7.39E−06**	1.00E+00	1.00E+00	1.00E+00	1.00E+00	9.97E−01	/
F6	Mean	5.89E+06	1.90E+07	1.68E+07	1.58E+01	1.63E+01	8.14E+04	4.11E+04	2.54E+06	0.00E+00
	Std	5.43E+06	2.12E+06	6.08E+06	7.30E−01	9.69E−01	2.84E+05	2.05E+05	7.02E+06	0.00E+00
	p-value	**6.54E−06**	**6.54E−06**	**6.54E−06**	**5.56E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	1.25E−01	/
F7	Mean	1.47E+05	4.88E+10	3.42E+10	2.79E+04	1.17E+04	1.23E+05	1.26E+10	0.00E+00	3.59E+09
	Std	2.47E+05	1.64E+10	5.19E+10	2.03E+04	3.96E+03	1.09E+05	1.48E+10	0.00E+00	3.68E+09
	p-value	1.00E+00	**6.54E−06**	**2.04E−02**	1.00E+00	1.00E+00	1.00E+00	**5.72E−03**	1.00E+00	/
F8	Mean	1.27E+08	8.23E+08	3.10E+10	2.78E+07	8.06E+05	7.50E+06	3.72E+07	5.88E+07	6.39E+07
	Std	1.52E+08	1.92E+08	6.90E+10	3.19E+07	1.63E+06	1.84E+07	3.47E+07	1.10E+08	2.13E+08
	p-value	**2.12E−04**	**7.39E−06**	**9.42E−06**	1.18E−01	9.95E−01	9.92E−01	1.35E−01	1.80E−01	/
F9	Mean	1.01E+08	1.69E+09	1.05E+09	3.65E+07	4.76E+07	1.02E+07	3.40E+08	5.49E+07	1.23E+02
	Std	9.09E+06	2.54E+08	1.13E+09	1.49E+07	5.30E+07	3.84E+06	2.67E+08	8.74E+06	1.16E+02
	p-value	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	/
F10	Mean	4.07E+03	5.19E+03	4.30E+03	3.33E+03	3.13E+03	2.59E+03	4.90E+03	1.82E+03	1.65E+03
	Std	1.26E+03	1.72E+03	1.77E+03	1.92E+02	1.68E+02	1.48E+02	6.37E+02	7.79E+01	5.15E+01
	p-value	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**7.39E−06**	/
F11	Mean	9.98E+01	2.00E+02	1.91E+02	2.64E+01	2.51E+01	2.69E+01	2.75E+01	8.10E+01	3.52E−02
	Std	1.01E+02	2.24E+00	3.56E+01	2.95E+00	2.72E+00	2.64E+00	3.18E+00	3.00E+01	1.76E−01
	p-value	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.53E−06**	**6.54E−06**	**6.54E−06**	**6.53E−06**	**6.54E−06**	/
F12	Mean	9.14E+03	8.68E+05	4.76E+05	3.21E+04	2.44E+04	3.53E+04	5.07E+04	3.97E+02	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	7.77E+01	0.00E+00
	p-value	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.53E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	/
F13	Mean	5.44E+03	3.24E+04	8.62E+04	2.89E+07	1.29E+04	9.06E+04	1.29E+07	3.22E+03	1.09E+03
	Std	2.76E+03	2.61E+04	1.95E+05	1.57E+07	4.34E+03	6.11E+04	7.36E+06	3.86E+03	4.21E+02
	p-value	**6.54E−06**	**6.54E−06**	**1.92E−05**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**1.29E−03**	/
F14	Mean	3.00E+08	3.62E+09	2.22E+09	2.10E+07	2.14E+07	2.24E+07	5.35E+09	3.12E+07	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	2.64E+06	0.00E+00
	p-value	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.53E−06**	**6.54E−06**	/
F15	Mean	1.30E+04	1.17E+04	1.10E+04	2.88E+03	2.84E+03	2.84E+03	3.22E+03	1.99E+03	1.92E+03
	Std	2.18E+02	2.05E+03	2.77E+03	2.76E+02	1.86E+02	2.65E+02	4.17E+02	8.65E+01	8.28E+01
	p-value	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**3.86E−03**	/
F16	Mean	2.02E+02	3.99E+02	3.62E+02	1.97E+01	1.93E+01	1.87E+01	1.91E+01	8.59E+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	3.96E+01	0.00E+00
	p-value	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	/
F17	Mean	7.47E+04	1.79E+06	9.71E+05	7.76E+00	7.08E+00	1.49E+01	1.24E+02	4.89E−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	1.11E−02	0.00E+00
	p-value	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	/
F18	Mean	1.44E+04	1.07E+05	7.77E+04	2.01E+10	1.15E+03	4.10E+09	1.23E+11	2.56E+03	1.42E+04
	Std	1.27E+04	2.68E+04	1.75E+05	4.82E+09	1.65E+02	1.83E+09	1.45E+10	1.22E+03	9.50E+03
	p-value	4.79E−01	**6.54E−06**	**3.85E−04**	**6.54E−06**	1.00E+00	**6.54E−06**	**6.54E−06**	1.00E+00	/

(continued on next page)

Table 2 *(continued)*

Functions		DECC-D	MLCC	DECC-DML	DECC-DG	DECC-I	CBCC-1	CBCC-2	CC-GDG-CMAES	NMMO-AFC-CC
F19	Mean	1.59E+06	2.96E+06	2.70E+06	9.01E+05	8.95E+05	9.12E+05	9.11E+05	2.99E+06	5.12E+03
	Std	1.32E+06	4.29E+05	3.37E+06	6.14E+04	6.24E+04	7.11E+04	6.02E+04	9.64E+05	7.15E+02
	p-value	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	/
F20	Mean	2.27E+03	1.75E+05	5.42E+03	6.53E+08	1.67E+07	1.41E+07	6.97E+09	1.02E+03	1.15E+03
	Std	2.44E+02	2.08E+05	1.46E+04	6.71E+08	3.30E+07	1.96E+07	1.12E+09	1.06E+02	9.11E+01
	p-value	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	**6.54E−06**	1.00E+00	/
Singnificant better No.		16 vs 2	19 vs 1	19 vs 1	16 vs 2	15 vs 4	16 vs 3	17 vs 1	12 vs 5	Overall Best No. 13

Table 3
Summary of statistical significant comparisons 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-D	~	–	+	+	+	+	–	+	+	+	+	+	+	+	+	+	+	~	+	+
MLCC	+	–	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
DECC-DML	+	–	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
DECC-DG	+	~	+	+	–	+	–	~	+	+	+	+	+	+	+	+	+	+	+	+
DECC-I	+	~	+	+	–	+	–	–	+	+	+	+	+	+	+	+	+	–	+	+
CBCC–1	+	~	+	+	–	+	–	–	+	+	+	+	+	+	+	+	+	+	+	+
CBCC–2	+	~	+	+	–	+	+	~	+	+	+	+	+	+	+	+	+	+	+	+
CC-GDG-CMAES	~	–	+	+	–	~	–	~	+	+	+	+	+	+	+	+	+	–	+	–

Table 4
Summary of the compared metaheuristic algorithms.

Algorithm	Description
2S-Ensemble [45]	Ranked the second in CEC'10 competition. A two-stage based ensemble optimization evolutionary algorithm. The search procedure is divided into two stages: 1) global shrinking with EDA based-on mixed Gaussian and Catchy models; 2) local exploration with three randomly selected optimizers.
DECC-G [1]	The base line CC algorithm with random grouping and SaNSDE optimizer.
jDEsps [46]	Ranked the second in CEC'12 competition. A self-adaptive differential evolution algorithm with a small and varying population size.
CC-CMA-ES [7]	Ranked the third in CEC'13 and CEC'15 competitions. Extending CMA-ES under the CC framework by extracting a subspace Gaussian distribution from the global Gaussian distribution for subspace sampling.
MA-SW-Chains [47]	Winner of CEC' 10 competition. It adopts a concept of local search chain in which each individual a local search intensity so that the most promising individuals may exploit with higher local search intensity. The Solis & Wets algorithm (SW) [48] is adopted as the local search optimizer.
MOS-CEC2012 [49]	Winner of CEC'12 competition. It is a hybrid approach combining the SW and the first local search method of the multiple trajectory search algorithm (MTS-LS1) [50] within a multiple offspring sampling (MOS) framework.
MOS-CEC2013 [51]	Winner of CEC'13 and CEC'15 competitions. Similar to MOS-CEC2012, but GA, SW and a variant of MTS-LS1, named MTS-LS1-Reduced, are combined within the framework of MOS.

Table 5
Statistical comparison results according to the Friedman average ranking and Formula 1 ranking tests with the best performing algorithm highlighted. A smaller Friedman ranking or larger Formula 1 ranking indicates better performance.

Algorithm name	Friedman ranking	Formula 1 ranking
2S-Ensemble	4.70	230
DECC-G	7.10	118
jDEsps	3.40	296
CC-CMA-ES	5.45	185
MA-SW-Chains	4.40	227
MOS-CEC2012	4.38	261
MOS-CEC2013	3.38	303
NMMO-AFC-CC	**3.20**	**340**

4.2.3. Constructing collaborators using the AFC

As mentioned in Section 3, the AFC is used to select a small number of collaborators (i.e., $\mathbf{Collab}(i)$) from the information pool $\mathbf{POOL}(i)$, which is a collection of collaborative information $\mathbf{CI}(j), j \neq i, j \in M$, provided by the other sub-components. Here, we let $\mathbf{POOL}(i) = \{\mathbf{CI}(j)\}, i, j \in M, i \neq j$. Note that the elements in $\mathbf{CI}(i) = \{\mathbf{S}_{nmmo}(i).\ \mathbf{OPT}, BI(i)\}, i \in M$ are provided in terms of whole solutions. In another words, each of them is provided together with the collaborative solution context that was identified during the fitness evaluation. Given the $\mathbf{POOL}(i)$, the collaborators that are used to conduct *best-of-N* fitness evaluation can be obtained by applying the AFC according to the corresponding problem domain of the *i*th sub-component:

$$\mathbf{Collab}(i) = AFC(\mathbf{POOL}(i), \mathcal{G}(i)). \tag{6}$$

Although the informative collaborators in $\mathbf{Collab}(i)$ are positive for the information compensation, it is not necessary to always use all of them to evaluate a given solution. Actually, an increased number of collaborators is only positive at the early co-evolutionary stage of the main optimizer (CMA-ES), and this number should be dynamically adjusted during the co-evolution process. The empirical study in [40] indicates that it appears to be beneficial to vary the number of collaborators with time to save computational resources. It is more reasonable to use more collaborators in the early CC stage, but only use

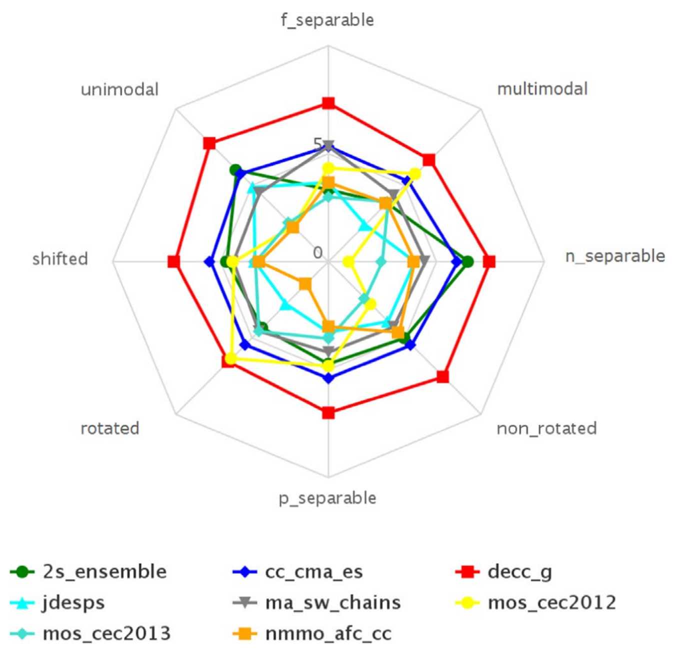

Fig. 2. Radar chart of the average Friedman ranking for different properties of the test functions.

a few collaborators in later generations. Our pilot experiments also showed that when evaluating a co-evolutionary population (each individual conducts a *best-of-N* evaluation), the index of the most frequently used element in **Collab**(i) tends to become gradually fixed as the co-evolution proceeds. These findings imply that we can further reduce the number of collaborators to 1 after a certain number of co-evolutionary cycles from the moment that NMMO is triggered. Accordingly, in a certain cycle after the NMMO procedure, if the index of the most frequently used element has not changed for N_l cycles, only such an element is used to conduct the fitness evaluation from then on until the NMMO is re-triggered.

5. Experimental results

In this section, two sets of experiments are carried out on the LSGO benchmark functions launched during CEC' 10 and CEC' 13, respectively. The effectiveness of the NMMO-AFC-CC is justified by comparing it with 16 state-of-the-art EAs for the LSGO.

5.1. General experimental settings

Here, the general settings are given for the experiments. To conduct a statistically significant comparison, each algorithm is executed for 25 independent runs for all cases. In each run, the algorithm terminates when the number of fitness evaluations (FEs) exceeds 3×10^6.

The general settings of the NMMO-AFC-CC are as follows. A state-of-the-art grouping method called differential grouping (DG) [9] is adopted to obtain the problem decomposition $\mathcal{G}$ in advance. In the experiments (unless otherwise specified) the controlling parameter of the DG is set to 10^{-6} for all corresponding algorithms, as suggested in [9]. Thus, the actual number of FEs equals 3×10^6 minus the FEs consumed in the DG procedure. NG_{nmmo} and N_l are empirically set to 100 and 10, respectively. To avoid an exhaustive search within a very small region for the CMA-ES optimizer, only 10% of the improvement is recognized as a better performance. The other algorithmic parameters of the NMMO and CMA-ES are set to the default values provided in [36,39].

5.2. Performance comparison with the CEC' 10 LSGO benchmark functions

A set of experiments are carried out to statistically compare the NMMO-AFC-CC with 8 state-of-the-art CC algorithms on the 20 benchmark test functions launched during the CEC' 10 LSGO competition. Those functions can be generally categorized into four categories: fully separable ($F1 \sim F3$), partially separable with single non-separable and separable groups ($F4 \sim F8$), partially separable with 10 non-separable and single separable groups ($F9 \sim F13$), partially separable with 20 non-separable groups ($F14 \sim F18$), and fully separable ($F19 \sim F20$) functions. The formula details can be seen in [41]. The compared CC algorithms are briefly described in Table 1. All the experimental results are collected either from a comprehensive comparison study [42] or by running the open source code related to the original papers.

The statistical results, including the values of the mean, standard deviation, and significant comparison, are given in Table 2. Note that the significant comparisons are conducted using a tailed Wilcoxon singed ranks test with a significant level of 0.05. If a *p-value* is smaller than 0.05, the NMMO-AFC-CC performs significantly better than the corresponding algorithm. Table 2 shows that the NMMO-AFC-CC outperforms the compared algorithms on most of the test functions and performs the best on 13 functions. This can be more obviously observed on the partially separable functions ($F9 \sim F17$) with several non-separable sub-components. This is because in those cases, the NMMO- and AFC-based information compensation is more likely to benefit CC algorithms.

The last row in Table 2 shows the number of significantly better results obtained by the NMMO-AFC-CC and each compared algorithm, respectively. In addition, more details about the significant comparisons are given in Table 3. These results show that the NMMO-AFC-CC obtains a significantly better result in 81.25% of the comparison cases; this is in contrast to the 11.875% rate when the NMMO-AFC-CC performs significantly worse.

According to the summary reports for the competitions at CEC' 10 to CEC' 15, all of the winners were metaheuristic algorithms. Actually, with the effective problem decomposition provided by the DG, the NMMO-AFC-CC performs very competitively when compared with some powerful metaheuristic algorithms. In the following set of experiments, the NMMO-AFC-CC is compared to 7 metaheuristic algorithms (summarized in Table 4), which performed competitively in the series of CEC LSGO competitions from 2010 to 2015. We conduct the comparison using an online benchmark result repository named MIDAS[4] to obtain a wide range of statistical results.

The overall statistical comparison results are given in Table 5 using the Friedman average ranking and Formula 1 ranking tests, respectively. The proposed NMMO-AFC-CC obtains the best overall performance according to both ranking tests, followed by the MOS-CEC2013, which was the winner of the most recent CEC competitions for the LSGO. To further elucidate how the algorithms perform with regard to the different properties of the test functions, a radar chart of the Friedman average ranking is given in Fig. 2. It can be seen that the proposed NMMO-AFC-CC performs the best on the functions that have partially separable, rotated, shifted, and unimodal properties. On such functions, the NMMO-AFC-CC has the corresponding average Friedman ranking values of 3.00, 1.44, 3.20, and 2.25 respectively. The second best Friedman average ranking values are 3.27, 2.78, 3.38, and 2.31 for jDEsps, jDEsps, MOS-CEC2013, and MOS-CEC2012, respectively. In contrast, the NMMO-AFC-CC performs relatively poorly on non-separable functions since there is no means of conducting information compensation on non-separable problems. The NMMO-AFC-CC does not perform well on non-rotated functions.

[4] http://vps128.cesvima.upm.es/

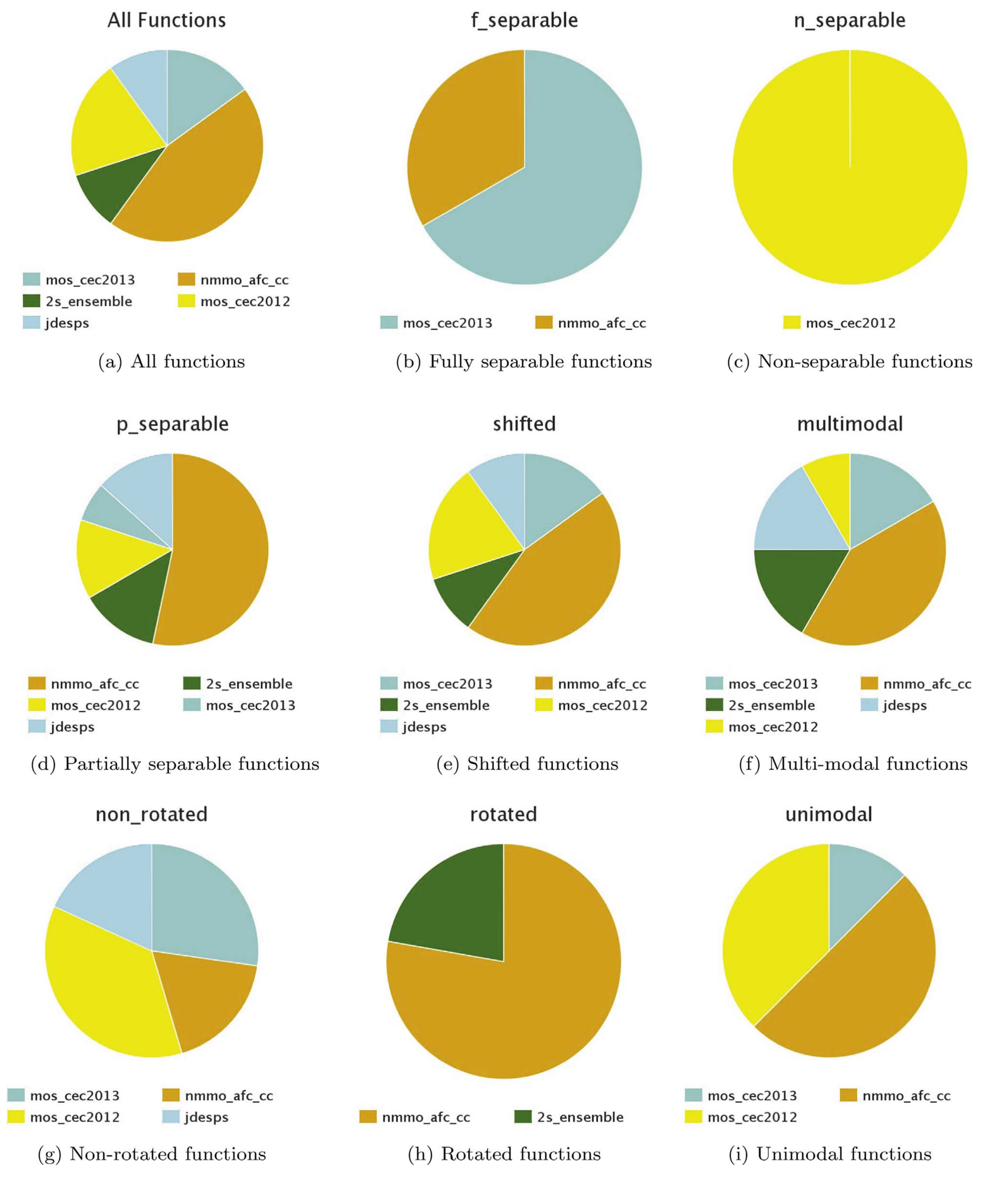

Fig. 3. Pie charts that describe the proportion of the functions for which each algorithm obtains the best results in terms of the mean error, (a) All functions, (b) Fully separable functions, (c) Non-separable functions, (d) Partially separable functions, (e) Shifted functions, (f) Multi-modal functions, (g) Non-rotated functions, (h) Rotated functions, (i) Unimodal functions.

This is because the rotation invariance property of the main optimizer (i.e. the CMA-ES) is more beneficial for solving rotated functions.

Fig. 3 provides another perspective to compare the algorithms by computing the proportion of the functions for which each algorithm performs the best according to the mean error (the values of the global optima are provided in [41]). The proposed NMMO-AFC-CC performs the best on 45% of the test functions, and MOS-CEC2012 and MOS-CEC2013 has the second and third places with percentages of 20% and 15%, respectively. The NMMO-AFC-CC wins the largest pie section on rotated functions (77.8%), partially separable functions (53.3%), unimodal functions (50%), shifted functions (45%), and multi-modal functions (41.7%). However, the NMMO-AFC-CC never wins on the

Table 6
Performance comparisons with L-SHADE on CEC'10 LSGO benchmark functions. The p-values are obtained using tailed Wilcoxon signed ranks test with a significant level of 0.05 for the alternative hypothesis that NMMO-AFC-CC performs better than the L-SHADE. A highlighted mean value indicate that the corresponding algorithm performs significantly better.

Functions		L-SHADE	NMMO-AFC-CC	p-value
F1	Mean	2.78E+07	**0.00E+00**	6.51E−06
	Std	2.39E+06	0.00E+00	
F2	Mean	**1.59E+03**	4.19E+03	1.00E+00
	Std	2.11E+03	1.28E+03	
F3	Mean	2.18E+00	**0.00E+00**	6.51E−06
	Std	1.01E−01	0.00E+00	
F4	Mean	1.11E+10	**1.77E+06**	6.54E−06
	Std	1.23E+09	1.05E+06	
F5	Mean	**9.34E+07**	1.56E+08	9.99E−01
	Std	7.44E+06	8.74E+07	
F6	Mean	2.02E+01	**0.00E+00**	6.51E−06
	Std	3.11E−02	0.00E+00	
F7	Mean	**3.76E+05**	3.59E+09	1.00E+00
	Std	3.67E+04	3.68E+09	
F8	Mean	2.19E+07	6.39E+07	5.64E−01
	Std	6.12E+05	2.13E+08	
F9	Mean	1.25E+08	**1.23E+02**	6.54E−06
	Std	5.61E+06	1.16E+02	
F10	Mean	**6.83E+02**	1.65E+03	1.00E+00
	Std	3.41E+01	5.15E+01	
F11	Mean	6.15E+01	**3.52E-02**	6.43E−06
	Std	4.10E+00	1.76E−01	
F12	Mean	4.37E+04	**0.00E+00**	6.39E−06
	Std	4.14E+03	0.00E+00	
F13	Mean	2.21E+05	**1.09E+03**	6.54E−06
	Std	2.41E+04	4.21E+02	
F14	Mean	2.80E+08	**0.00E+00**	6.51E−06
	Std	1.16E+07	0.00E+00	
F15	Mean	**6.87E+02**	1.92E+03	1.00E+00
	Std	2.56E+01	8.28E+01	
F16	Mean	9.11E+01	**0.00E+00**	6.51E−06
	Std	9.25E+00	0.00E+00	
F17	Mean	1.74E+05	**0.00E+00**	6.51E−06
	Std	5.98E+03	0.00E+00	
F18	Mean	3.71E+04	**1.42E+04**	1.35E−05
	Std	1.24E+04	9.50E+03	
F19	Mean	7.17E+05	**5.12E+03**	6.54E−06
	Std	2.59E+04	7.15E+02	
F20	Mean	5.13E+03	**1.15E+03**	6.54E−06
	Std	2.91E+02	9.11E+01	

Table 7
Performance comparisons on CEC'13 LSGO benchmark functions. The p-values are obtained using tailed Wilcoxon signed ranks test with a significant level of 0.05 for the alternative hypothesis that NMMO-AFC-CC performs better than DECC-DG. The results of MOS-CEC2013 is obtained directly from the original paper [51]. A highlighted *p-value* means NMMO-AFC-CC performs significantly better than DECC-DG.

Functions		DECC-DG	NMMO-AFC-CC	MOS-CEC2013
F1	Mean	4.18E+03	0.00E+00	0.00E+00
	Std	1.42E+04	0.00E+00	0.00E+00
	p-value	**6.54E-06**	/	N/A
F2	Mean	1.24E+04	1.39E+03	8.36E+02
	Std	6.41E+02	6.78E+01	8.32E+02
	p-value	**6.54E-06**	/	N/A
F3	Mean	2.14E+01	2.09E+01	9.10E+13
	Std	1.13E−02	1.13E−02	9.17E+13
	p-value	**6.54E-06**	/	N/A
F4	Mean	7.31E+10	5.29E+10	1.56E+08
	Std	2.75E+10	3.28E+10	1.74E+08
	p-value	**1.46E-02**	/	N/A
F5	Mean	5.77E+06	2.29E+06	6.79E+06
	Std	3.30E+05	5.22E+05	6.94E+06
	p-value	**6.54E-06**	/	N/A
F6	Mean	1.06E+06	1.06E+06	1.39E+05
	Std	1.09E+03	1.22E+03	1.48E+05
	p-value	1.00E+00	/	N/A
F7	Mean	4.53E+08	3.96E+08	1.62E+04
	Std	1.77E+08	5.53E+08	1.62E+04
	p-value	1.18E−01	/	N/A
F8	Mean	3.81E+15	5.87E+13	8.08E+12
	Std	1.46E+15	5.42E+13	8.00E+12
	p-value	**6.54E−06**	/	N/A
F9	Mean	6.37E+08	1.54E+08	3.87E+08
	Std	4.71E+07	3.15E+07	3.83E+08
	p-value	**6.54E−06**	/	N/A
F10	Mean	9.41E+07	9.49E+07	1.18E+06
	Std	2.99E+05	3.08E+05	9.02E+05
	p-value	1.00E+00	/	N/A
F11	Mean	9.23E+11	7.02E+10	4.48E+07
	Std	7.30E+09	5.19E+10	5.22E+07
	p-value	**6.54E−06**	/	N/A
F12	Mean	1.73E+11	1.30E+03	2.46E+02
	Std	1.55E+10	1.65E+02	2.47E+02
	p-value	**6.54E−06**	/	N/A
F13	Mean	2.08E+10	1.83E+10	3.30E+06
	Std	4.47E+09	6.78E+09	3.40E+06
	p-value	5.04E−02	/	N/A
F14	Mean	2.64E+10	1.03E+07	2.42E+07
	Std	1.61E+10	5.55E+06	2.56E+07
	p-value	**6.54E−06**	/	N/A
F15	Mean	1.17E+15	2.30E+06	2.38E+06
	Std	0.00E+00	2.36E+05	2.35E+06
	p-value	**6.54E−06**	/	N/A

non-separable functions where it actually does not run in a CC fashion with a single group of decision variables.

Since the we focus on real-valued single objective optimization problems, the success-history based adaptive DE [52] using linear population size reduction (L-SHADE) [53] is further used to compare with the NMMO-AFC-CC. The L-SHADE was a first ranked method at the CEC2014 Competition on Real-Parameter Single Objective Optimization. Since the L-SHADE was not test on benchmarks functions for CEC LSGO competitions in the original work and the corresponding experimental data is unavailable on MIDAS, here we conduct a separate experiment using the source code provided by the authors of [53]. The performance of the L-SHADE and NMMO-AFC-CC is statistically compared according to the Wilcoxon signed ranks test with a significant level of 0.05.

As shown in Table 6, the proposed NMMO-AFC-CC performs significantly better on 14 functions while the compared L-SHADE performs significantly better on 5 functions. Interestingly, among the 5 functions that the NMMO-AFC-CC is outperformed, $F2$, $F5$, $F10$ and $F15$ are constructed based on the Rastrigin function. This indicates the clear superiority of L-SHADE on Rastrigin functions.

5.3. Performance comparison on the CEC'13 LSGO benchmark functions

In the following experiments, the proposed NMMO-AFC-CC is compared on the 15 benchmark functions launched during the CEC'13 LSGO competition. Comparing with the 20 test functions launched during the CEC'10 LSGO competition, four new features were introduced [54]: non-uniform sub-component sizes, imbalance in the contribution of the sub-components, functions with overlapping sub-components, and new transformations to the base functions. These features are very challenging for problem decomposition. As reported in [42], the DG can only accurately decompose $F5$ and $F10$. This is even worse for the GDG, according to our pilot experiments. For example, it groups all variables into a single non-separable group when conducting the decomposition on $F3$ (fully separable), $F11$ (partially separable), and $F12 \sim F14$ (overlapping). Thus, in the following experiments, only the DG-based decomposition is considered.

The statistical results of the comparison between the proposed NMMO-AFC-CC and the DECC-DG are given in Table 7 with the results of the MOS-CEC2013 (the winner of CEC'13 and CEC'15 LSGO competitions) provided as a reference. Like the above experiments, the tailed Wilcoxon singed ranks test with a significant level of 0.05 is used to make a comparison. The proposed NMMO-AFC-CC performs significantly better than the DECC-DG on 11 out of 15 functions and is beaten only on $F10$. However, compared to the MOS-CEC13, due to the unsatisfying problem decomposition, the NMMO-AFC-CC failed to perform competitively as it did when optimizing the benchmark functions launched at CEC'10.

6. Conclusions

The *divide-and-conquer* problem-solving manner endows CC algorithms with a very promising means for the LSGO. Two crucial aspects must be carefully considered to make full use of the strength of CC algorithms: the problem decomposition and optimization procedure. In contrast to the problem decomposition, which has been relatively well studied for the LSGO, little research has contributed to the CC framework in the context of the LSGO. Although the algorithms from the literature use some promising problem decomposition approaches, many state-of-the-art large-scale CC algorithms for the LSGO (such as DECC-D, MLCC, CC-CMA-ES, DECC-DG, and CC-GDG-CMAES) still adopt a very simple CC framework. In such framework, only the single best collaborative solution is provided by each sub-component. Indeed, early works about CC algorithms in the context of a low-dimensional problem domain argued that more informative collaborators are positive for helping CC algorithms address the inherent pathological behaviors caused by problem decomposition. Although some approaches have been proposed to conduct information compensation, they are not scalable for the LSGO with a number of sub-components. A combinational explosion may be induced, which leads to a significant FE consumption when assessing a certain individual.

To increase the information compensation, a niching-based multimodal optimization (NMMO) procedure was introduced into the canonical CC framework to increase the number of informative collaborators when the main optimizing procedure becomes stuck. A number of current optima (local and global) of each corresponding sub-component were identified as informative collaborators. Those collaborators were positive for the sub-components and allowed them to make an accurate estimation of the joint landscape. Thus, this procedure is more likely to achieve a global optimization. In addition to the NMMO-based information compensation, an efficient fast clustering method was extended to cluster the collated collaborators without manually selecting the controlling parameters. The benefits from the resultant adaptive fast clustering's (AFC) number of collaborators that were used to evaluate a given individual were dramatically reduced and therefore saved computational resources. To further save computational resources, a simple but effective scheme was proposed to identify the most frequently used collaborators. These collaborators were used instead of all the collaborators to conduct the fitness evaluation from a certain moment on. Therefore, in summary, the resultant NMMO-AFC-CC increases the information compensation while avoiding a large number of fitness evaluations.

As for the experimental studies, two benchmark suites launched at the CEC'10 and CEC'13 LSGO competitions have been used to verify the effectiveness of the NMMO-AFC-CC. 16 state-of-the-art EAs have been compared with the NMMO-AFC-CC. According to the experimental results on the CEC'10 test functions for the comparison with the 8 large-scale CC algorithms, the NMMO-AFC-CC performs significantly better (according to the Wilcoxon signed ranks) in 81.25% of the comparison cases and performs the best on 13 out of 20 functions. In addition, when compared with 7 large-scale metaheuristic algorithms, the NMMO-AFC-CC obtains the overall best performance and is superior on most of the features of the test functions according to both the Friedman and Formula 1 ranking tests. When comparing with the L-SHADE the NMMO-AFC-CC also shows superiority on most of the test functions excepting those constructed based on the Rastrigin function. The NMMO-AFC-CC has also been compared with the DECC-DG on the 15 CEC'13 test functions with the same problem decomposition method (i.e., DG). The statistical results indicate that the proposed algorithm significantly outperforms the DECC-DG on 11 out of 15 test functions.

An interesting topic for future work is the design of a learning-based online problem decomposition and its incorporation into an information-compensation-based large-scale CC algorithm. In addition, we hope to conduct a set of very comprehensive experiments to analyze sensitivity of the NMMO-AFC-CC to different configurations in the future work.

Acknowledgement

This work has been supported by National Nature Science Foundation of China (No. 61105068 and 61473233), Fundamental Research Funds for the Central Universities of China under Grant 3102016ZY007.

References

[1] Z. Yang, K. Tang, X. Yao, Large scale evolutionary optimization using cooperative coevolution, Inf. Sci. 178 (15) (2008) 2985–2999.
[2] M.N. Omidvar, X. Li, Z. Yang, X. Yao, Cooperative co-evolution for large scale optimization through more frequent random grouping, in: Proceedings of The IEEE Congress on Evolutionary Computation (CEC 10), 2010, pp. 1–8.
[3] Z. Yang, K. Tang, X. Yao, Multilevel cooperative coevolution for large scale optimization, in: Proceedings of The IEEE Congress on Evolutionary Computation (CEC 08), 2008, pp. 1663–1670.
[4] X. Li, X. Yao, Cooperatively coevolving particle swarms for large scale optimization, IEEE Trans. Evolut. Comput. 16 (2) (2012) 210–224.
[5] A. Kabán, J. Bootkrajang, R.J. Durrant, Toward large-scale continuous eda: a random matrix theory perspective, Evolut. Comput. 24 (2) (2016) 255–291.
[6] M.N. Omidvar, X. Li, X. Yao, Cooperative co-evolution with delta grouping for large scale non-separable function optimization, in: Proceedings of The IEEE Congress on Evolutionary Computation (CEC 10), 2010, pp. 1–8.
[7] J. Liu, K. Tang, Scaling up covariance matrix adaptation evolution strategy using cooperative coevolution, in: Intelligent Data Engineering and Automated Learning (IDEAL 13), Springer, 2013, pp. 350–357.
[8] W. Chen, T. Weise, Z. Yang, K. Tang, Large-scale global optimization using cooperative coevolution with variable interaction learning, in: Proceedings of Parallel Problem Solving from Nature (PPSN XI), Springer, 2010, pp. 300–309.
[9] M.N. Omidvar, X. Li, Y. Mei, X. Yao, Cooperative co-evolution with differential grouping for large scale optimization, IEEE Trans. Evolut. Comput. 18 (3) (2014) 378–393.
[10] Y. Mei, M.N. Omidvar, X. Li, X. Yao, A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization, ACM Trans. Math. Softw. 42 (2) (2016) (13:1–24).

[11] R.P. Wiegand, An analysis of cooperative coevolutionary algorithms, (Ph.D. thesis), George Mason University, Fairfax, VA, USA, 2004.
[12] L. Panait, Theoretical convergence guarantees for cooperative coevolutionary algorithms, Evolut. Comput. 18 (4) (2010) 581–615.
[13] L. Panait, S. Luke, Selecting informative actions improves cooperative multiagent learning, in: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, ACM, 2006, pp. 760–766.
[14] X. Peng, K. Liu, Y. Jin, A dynamic optimization approach to the design of cooperative co-evolutionary algorithms, Knowl.-Based Syst. 109 (2016) 174–186.
[15] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.
[16] S. Wang, D. Wang, C. Li, Y. Li, Comment on Clustering By Fast Search And Find Of Density Peaks. http://arXiv:1501.04267arXiv:1501.04267.
[17] I. Barany, V. Vu, Central limit theorems for gaussian polytopes, Ann. Probab. (2007) 1593–1621.
[18] K.A. De Jong, Analysis of behavior of a class of genetic adaptive systems, (Ph.D. thesis), University of Michigan, Ann Arbor, MI, USA, 1975.
[19] S.W. Mahfoud, Niching methods for genetic algorithms, Technical Report IlliGAL Report No. 95001, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, USA, 1995.
[20] O.J. Mengshoel, D.E. Goldberg, Probabilistic crowding: Deterministic crowding with probabilistic replacement, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 99), 1999, pp. 409–416.
[21] D.E. Golberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addion wesley, 1989.
[22] X. Yin, N. Germay, A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization, in: Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms, Springer, 1993, pp. 450–457.
[23] W. Gao, G.G. Yen, S. Liu, A cluster-based differential evolution with self-adaptive strategy for multimodal optimization, IEEE Trans. Cybern. 44 (8) (2014) 1314–1327.
[24] G. R. Harik, Finding multimodal solutions using restricted tournament selection., in: Proceedings of the Internationbal Conference on Genetic Algorithm (ICGA 95), 1995, pp. 24–31.
[25] R. Roy, I.C. Parmee, Adaptive restricted tournament selection for the identification of multiple sub-optima in a multi-modal function, in: Proceedings of the AISB Workshop on Evolutionary Computing, Springer, 1996, pp. 236–256.
[26] A. Pétrowski, A clearing procedure as a niching method for genetic algorithms, in: Proceedings of The IEEE Conference on Evolutionary Computation (CEC 96), 1996, pp. 798–803.
[27] J.-P. Li, M.E. Balazs, G.T. Parks, P.J. Clarkson, A species conserving genetic algorithm for multimodal function optimization, Evolut. Comput. 10 (3) (2002) 207–234.
[28] B.-Y. Qu, P.N. Suganthan, J.-J. Liang, Differential evolution with neighborhood mutation for multimodal optimization, IEEE Trans. Evolut. Comput. 16 (5) (2012) 601–614.
[29] B.-Y. Qu, P.N. Suganthan, S. Das, A distance-based locally informed particle swarm model for multimodal optimization, IEEE Trans. Evolut. Comput. 17 (3) (2013) 387–402.
[30] I.L. Schoeman, A.P. Engelbrecht, A novel particle swarm niching technique based on extensive vector operations, Nat. Comput. 9 (3) (2010) 683–701.
[31] A. Basak, S. Das, K.C. Tan, Multimodal optimization using a biobjective differential evolution algorithm enhanced with mean distance-based selection, IEEE Trans. Evolut. Comput. 17 (5) (2013) 666–685.
[32] S. Biswas, S. Kundu, S. Das, An improved parent-centric mutation with normalized neighborhoods for inducing niching behavior in differential evolution, IEEE Trans. Cybern. 44 (10) (2014) 1726–1737.
[33] A. Ahrari, K. Deb, M. Preuss, Multimodal optimization by covariance matrix self-adaptation evolution strategy with repelling subpopulations, Evolutionary Computation.
[34] Q. Yang, W.-N. Chen, Y. Li, C.P. Chen, X.-M. Xu, J. Zhang, Multimodal estimation of distribution algorithms, accepted by IEEE Transactions on Cybernetics.
[35] S. Das, S. Maity, B.-Y. Qu, P.N. Suganthan, Real-parameter evolutionary multi-modal optimization–a survey of the state-of-the-art, Swarm Evolut. Comput. 1 (2) (2011) 71–88.
[36] J.E. Fieldsend, Running up those hills: Multi-modal search with the niching migratory multi-swarm optimiser, in: Proceedings of The IEEE Congress on Evolutionary Computation (CEC 14), 2014, pp. 2593–2600.
[37] X. Li, A. Engelbrecht, M. Epitropakis, Results of the 2013 ieee cec competition on niching methods for multimodal optimization, in: Report presented at 2013 IEEE Congress on Evolutionary Computation Competition on Niching Methods for Multimodal Optimization, 2013.
[38] N. Hansen, A. Ostermeier, Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation, in: Proceedings of IEEE International Conference on Evolutionary Computation1996, pp. 312–317.
[39] N. Hansen, The CMA Evolution Strategy: a Tutorial, CoRR arXiv:1604.00772 URL ⟨http://arxiv.org/abs/1604.00772⟩.
[40] L. Panait, S. Luke, Time-dependent collaboration schemes for cooperative coevo-lutionary algorithms, in: Proceedings of the 2005 AAAI Fall Symposium on Coevolutionary and Coadaptive Systems, 2005.
[41] K. Tang, X. Li, P.N. Suganthan, Z. Yang, T. Weise, Benchmark functions for the cec 2010 special session and competition on large-scale global optimization, Tech. rep., Nature Inspired Computation and Applications Laboratory, 2009.
[42] M.N. Omidvar, X. Li, K. Tang, Designing benchmark problems for large-scale continuous optimization, Inf. Sci. 316 (2015) 419–436.
[43] Z. Yang, K. Tang, X. Yao, Self-adaptive differential evolution with neighborhood search, in: Proceedings of The IEEE Congress on Evolutionary Computation (CEC 08), 2008, pp. 1110–1116.
[44] M.N. Omidvar, X. Li, 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, Elsevier, 2015, pp. 1115–1122.
[45] Y. Wang, B. Li, Two-stage based ensemble optimization for large-scale global optimization, in: Proceedings of The IEEE Congress on Evolutionary Computation (CEC 10), 2010, pp. 1–8.
[46] J. Brest, A. Zamuda, I. Fister, M.S. Maučec, et al., Self-adaptive differential evolution algorithm with a small and varying population size, in: Proceddings of The IEEE Congress on Evolutionary Computation (CEC 12), 2012, pp. 1–8.
[47] D. Molina, M. Lozano, F. Herrera, Ma-sw-chains: Memetic algorithm based on local search chains for large scale continuous global optimization, in: Proceedings of The IEEE Congress on Evolutionary Computation (CEC 10), 2010, pp. 1–8.
[48] F.J. Solis, R.J.-B. Wets, Minimization by random search techniques, Math. Oper. Res. 6 (1) (1981) 19–30.
[49] A. LaTorre, S. Muelas, J.-M. Pena, Multiple offspring sampling in large scale global optimization, in: Proceedings of The IEEE Congress on Evolutionary Computation (CEC 12), 2012, pp. 1–8.
[50] L.-Y. Tseng, C. Chen, Multiple trajectory search for large scale global optimization, in: Proceedings of The IEEE Congress on Evolutionary Computation (CEC 08), 2008, pp. 3052–3059.
[51] A. LaTorre, S. Muelas, J.-M. Pena, Large scale global optimization: Experimental results with mos-based hybrid algorithms, in: Proceedings of The IEEE Congress on Evolutionary Computation (CEC 13), 2013, pp. 2742–2749.
[52] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differ-ential evolution, in: Proceedings of The IEEE Congress on Evolutionary Computation (CEC 13), 2013, pp. 71–78.
[53] R. Tanabe, A.S. Fukunaga, Improving the search performance of shade using linear population size reduction, in: Proceedings of The IEEE Congress on Evolutionary Computation (CEC 14), 2014, pp. 1658–1665.
[54] X. Li, K. Tang, M.N. Omidvar, Z. Yang, K. Qin, H. China, Benchmark functions for the cec 2013 special session and competition on large-scale global optimization, gene 7 (33), 2013, 8.