A Dynamic Global Differential Grouping for Large-Scale Black-Box Optimization

Shuai Wu[1], Zhitao Zou[2], Wei Fang[1]

[1] School of IoT Engineering, Jiangnan University
[2] School of Science, Jiangnan University

Abstract. Cooperative Co-evolution (CC) framework is an important method to tackle Large Scale Black-Box Optimization (LSBO) problem. One of the main step in CC is grouping for the decision variables, which affects the optimization performance. An ideal grouping result is that the relationship of decision variables in intra-group is stronger as possible and those in inter-groups is weaker as possible. Global Differential Grouping (GDG) is an efficient grouping method based on the idea of partial derivatives of multivariate functions, and it can automatically resolve the problem by maintaining the global information among variables. However, once the grouping result by GDG is determined, it will no longer be updated and will not be automatically adjusted with the evolution of the algorithm, which may affect the optimization performance of the algorithm. Therefore, based on GDG, a Dynamic Global Differential Grouping(DGDG) strategy is proposed for grouping the decision variables in this paper, which can update the grouping results with the evolution processing. DGDG works with Particle Swarm Optimization (PSO) algorithm in this paper, which is termed as CC-DGDG-PSO. The experimental results based on the LSBO benchmark functions from CEC'2010 show that DGDG algorithm can improve the performance of GDG.

Keywords: Large-scale Black-box Optimization, Cooperative co-evolution, Dynamic, Differential grouping

1 Introduction

Large Scale Global Optimization (LSGO) problems refer to the problems with a large number of decision variables to be optimized, usually thousands or even more [1]. In LSGO problems, there is a case where the objective function does not have an explicit analytical formula [3], which is usually called Large Scale Black-Box Optimization (LSBO) problem [4]. LSBO problems are often encountered in research and industrial fields [5]. For instance, in the optimization problem of fluid-based airplane wing shapes, there are more than 2500 variables need to be optimized [6, 7]. The optimization of LSBO problems is encountered two main issues:(1)the characteristics of black-boxes and the problem often have

complex features such as nonlinear, non-convex and non-differentiable; (2)large-scale decision variables.

Cooperative Co-evolution (CC)[8] , which is based on the idea of divide-and-conquer, is an algorithmic framework for solving LSBO problems by decomposing the problem into some sub-problems. Since the CC framework can decompose the LSBO problem into some sub-problems that the optimization algorithm can process, it has become an important method to solve the LSBO problem in recent years [1, 2]. But one of the key issues is how to divide the problem effectively with high accuracy[2, 9].

The research on the grouping strategy can divide into three categories: fixed grouping, stochastic grouping and grouping based on learning mechanism. Dividing the problem into some sub-problems with the pre-fixed groups is called fixed grouping strategy, i.e., the problem with n variables can divide into m sub-problems with s variables ($n = m \cdot s, 1 \leq s \leq \frac{n}{2}$)[10]. The fixed grouping strategy performed better on separable problems. However, the size of sub-problems and the composition of variables are all pre-determined , and the correlation between variables is not considered. Therefore, it usually shows weak optimization ability in nonseparable problems. In order to study the correlation between variables and place the associated variables as much as possible in the same sub-problem, the researchers proposed a stochastic grouping method in which variables in sub-problems are assigned in a random way and variables change with the evolution process. In [11, 12], Yang et al ,designed a decomposition strategy that randomly assigned variables to a fixed number of groups to increase the probability of the associated variables entering the same group. However, when the number of associated variables exceeds 5 in a group, the effect of this strategy becomes weak [13]. Later, Yang et al. proposed a multi-level co-evolutionary approach (MLCC) to solve the problem that the optimal number of groups in [12] is determined difficultly [14, 15], but the size of each group is still equal. Grouping based on learning mechanism, that is, before or during the implementation of the optimization algorithm, through the analysis of certain features to learn the relationship between variables so that grouping the variables clearly. Ray and Yao proposed a method of grouping variables according to the best fitness of the first 50% of individuals [16]. Later, Singh and Ray improved this method [17]. In order to improve the variable grouping method of MLCC [14], [18] proposed a new method of adaptively obtaining the size of variable groupings. J. Liu, K. Tang et al, have implemented problem decomposition based on the features of evolutionary algorithms [19, 20]. In [21], a grouping method based on the idea of partial derivative of multivariate functions is proposed to realize the automatic decomposition of the problem. This method obtains very accurate grouping results in most of the test functions of CEC'2010 [22]. However,to some extent, it depends on the pre-defined threshold, and the accuracy of problem decomposition will decrease with the increasing of the complexity of the problem (unevenness of sub-problems, large difference contribution of sub-problems, and the increase of the correlation between variables) [23]. In [5], Yi Mei proposed a GDG approach to reduce the dependency of the pre-defined threshold and ad-

dressed the missing relationship among variables [21]. However, there are some drawbacks in the decomposition of complex optimization problems and the judgment of the complexity of correlation between variables. And once the results of variable grouping are determined, they are no longer updated, which can not reflected the influence of evolution process on the correlation between variables.

In this paper, we propose a new GDG method named Dynamic GDG(D-GDG) to address the lack of GDG method, which can dynamically adjust the grouping of variables with process.

The rest of the paper is organised as follows: The definition of LSBO problem and CC framework are introduced in Section 2. Then, the introduction of GDG method and the proposed algorithm, named Dynamic GDG(D-GDG), is described in Section 3. the experimental studies are presented in Section 4. Finally, conclusions is described in Section 5.

2 The Defnition of LSBO Problem and CC Framework

2.1 The Defnition of LSBO Problem

Without loss of generality, a LSBO problem can be stated as follows:

$$min\ f(\mathbf{x}) \tag{1}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_D)$ is a D-dimensional *decision vector*,and each $x_i (i = 1, \ldots, D)$ is called a *decision variable*. D, the number of decision variables, is the dimension of search space. The real value of D is generally relatively large, usually in the hundreds to thousands. $f(\mathbf{x})$ is the *objective function* to be minimized.

2.2 CC Framework

The main step of CC framework can be stated as follows:

Step 1 **variables grouping**: Divide the original problem into multiple sub-problems. The method of dividing is more important. The bad dividing method will lead to strong coupling between groups and affect the optimization performance of the whole algorithm. Therefore, the design of the divide and conquer strategy is the most crucial step to improve the algorithm's performance.

Step 2 **the optimization of sub-problems**: To solve each sub-problem independently, this step should select the optimizer with significant performance. It can optimize the sub-problems in a round-robin fashion or independently optimize with the parallel program.

Step 3 **the merge of sub-problems**: Use the solution of the current sub-problem and the best solution of other sub-problems to synthesize a complete solution vector and evaluate the fitness of this solution vector.

The pseudocode of CC framework is stated as follows:

Algorithm 1 CC(f,lb,ub,D)

```
1: /* (Phase 1): Decomposition */
2: groups ← grouping(f,lb,ub,D);
3: /* (Phase 2): Optimization */
4: /* Initialization */
5: pop ← rand(popsize, D);
6: (best,bestval) ← min f(pop) ;
7: for i ← 1 to Cycle do
8:    for j ← 1 to size(groups) do
9:       indices ← groups[j];
10:      subpop ← pop[:,indices];
11:      /* Use sub-optimizer*/
12:      subpop ← optimizer(best, subpop, FE);
13:      pop[:, indicies]← subpop;
14:      (best, bestval) ← min(f(pop);
15:   end for
16: end for
```

3 Dynamic Global Differential Grouping

No matter what kind of decomposition method is adopted, the main goal of variable decomposition is to place the interrelated variables in the same group and separate the unrelated variables. Therefore, the core idea of variable decomposition is to find the correlation between variables. There are some related features such as *completely separable, completely nonseparable, partially separable, overlapping* between variables in LSBO problem. In a black-box environment, the information related to the problem what we have only know is the number of variables and its domain with the output of the objective function corresponding to different inputs. The relationship between the variables are completely unknown. Therefore, analyse the relationship between variables by evaluating the fitness of objective function is a viable option.

3.1 Global Differential Grouping

The GDG method is extended from the Differential Grouping (DG) method. Assume that there are optimization problems as follows:

$$F(x_1, x_2, x_3, x_4) = x_1^2 + x_2 x_3^{-3} + x_3 x_4 + x_4^2 \tag{2}$$

First, when the DG method detects the codependency of x_2 and x_3, the two variables are grouped together and the variables x_2 and x_3 are eliminated from the variable pool. Next, it is detected that x_3 and x_4 are interdependent and put in another group. So the interdependence between x_2 and x_4 can not be detected. Therefore, there are some drawback like this , missing some correlation variables which should have divided into a same group in some optimization problem. Mei proposed a GDG method based on DG method. Instead of the step of variable

elimination in DG method, Mei calculated all the difference of objective function fitness in the algorithm and obtained a complete difference matrix. Then every element of difference matrix compared with a sufficiently small threshold ϵ to get a correlation matrix containing only 0 and 1. Finally, the decomposition of the variables can be modelled as the computation of the connected components of the graph with the node adjacency matrix, which can be easily solved using *breadth-first search* or *depth-first search*[5]. The GDG method greatly improve the accuracy of grouping.

3.2 Dynamic GDG (D-GDG)

The GDG method is the same as the DG method in that all of them determine the variable grouping before algorithm is ran and the grouping result runs through the whole optimization process. Both methods do not consider the appropriate adjustment of the grouping of variables as the optimization process advances ,which may not adapt to the current distribution of solutions. In this paper, based on the basic idea of GDG method, the original data matrix that reflects the connection between two variables is derived from the general idea of partial derivation of multivariate functions. Through the standardization of the original data matrix and the establishment of the fuzzy relation matrix, the fuzzy clustering method is used to realize the dynamic clustering of variables. And through limiting the size of each variable group by setting the upper and lower bounds of the variable grouping scale, according to the state of the algorithm during operation, the grouping of variables is adaptively adjusted to promote the optimization of the problem.

The pseudocode of DGDG is described in Algorithm 2. An example of DGDG is given blow to explain DGDG algorithm. Firstly, the original difference matrix Λ obtained by DGDG algorithm using the method to compute the original difference matrix described in GDG algorithm. Assumed we have got the *Lambda* matrix shown in Eq. (3). Then, the Candidate threshold vector $\epsilon\_vec = (0, 1, 2, 4, 5)$ are obtained by sorting and de-duplicating each element in the Λ matrix. DGDG introduces a hyperparameter, α, which represents the number of fuzzy clustering, ie, the times of variables grouping in whole algorithm ran process. Assumed α is equal to 2 in our example. So the distance between two adjacent selected thresholds, *step* is equal to 2, the floor of the size of ϵ_vec divided by α. And all selected thresholds is vector $(5, 2, 0)$. Finally, the GDG method is used for each threshold one by one. For instance, assumed the threshold currently to be processed is 2. Then, a matrix Θ is obtained from Λ and $\epsilon(\epsilon \in \epsilon\_vec)$. The entry Θ_{ij} takes 1 if $\Lambda_{ij} > \epsilon$, and 0 otherwise. The Θ matrix is shown in Eq. (4). Considering Θ matrix as the adjacency matrix of a graph, we got two groups, $\{x_1, x_5\}$ and $\{x_2, x_3, x_4\}$.

$$\Lambda = \begin{pmatrix} 0\,1\,2\,1\,4 \\ 1\,0\,4\,5\,2 \\ 2\,4\,0\,2\,2 \\ 1\,5\,2\,0\,2 \\ 4\,2\,2\,2\,0 \end{pmatrix} \tag{3}$$

$$\Theta = \begin{pmatrix} 0\,0\,0\,0\,1 \\ 0\,0\,1\,1\,0 \\ 0\,1\,0\,0\,0 \\ 0\,1\,0\,0\,0 \\ 1\,0\,0\,0\,0 \end{pmatrix} \tag{4}$$

Algorithm 2 DGDG(f,lb,ub,D,α)

1: $\Lambda \leftarrow calcDiffMat(f, D, lb, ub)$ //computing the original difference matrix using the method described in paper [5]
2: $\epsilon\_vec \leftarrow unique(sort(diff(:)))$ //obtaining the candidate threshold vector
3: idx $\leftarrow size(\epsilon\_vec)$
4: step $\leftarrow floor(size(\epsilon\_vec)/\alpha)$
5: *Initialize the best solution, bestx, bestval*
6: **while** idx > 0 **do**
7: $\epsilon \leftarrow \epsilon\_vec(idx)$
8: $\Theta \leftarrow \Lambda > \epsilon$
9: groups $\leftarrow grouping(\Theta)$
10: (bestx,bestval) $\leftarrow sub\_optimizer(f, lb, ub, D, groups, bestx, bestval)$ //using sub-optimizer to find the best solution according to the current solution and current groups
11: idx $\leftarrow$ idx - step
12: **end while**

4 Experimental Results and Discussions

4.1 Experimental Settings

In order to show the effect of the proposed DGDG method for solving the LSBO problem, the proposed DGDG algorithm is evaluated on the CEC'2010 LSGO benchmark functions and use PSO optimal algorithm as sub-optimizer. The final results obtained by CC-DGDG-PSO are compared with CC-GDG-PSO proposed by Mei et al. The stop criterion $maxFEs = 3 \times 10^6$ is a commonly-used setting. Besides this parameter, there is only one parameter related to DGDG(Algo. (2))- α. Three different values $(3, 6, 9)$ for α are took to experiment on DGDG algorithm. All the compared algorithms were run 25 times independently to reduce the randomness of experiment.

The CEC'2010 LSGO benchmark functions which consist of 20 $1000-dimensional$ benchmark function (f_1 to f_{20}) can be divided into three categories, fully separable ($f_1 - f_3$), partially separable ($f_4 - f_{18}$), fully non-separable ($f_{19} - f_{20}$).

4.2 Results and Discussions

Table 1,2,3 respectively shows the minimal, median, mean and standard deviation of fitness values obtained by the 25 independent runs of the compared algorithms on fully separable functions($f_1 - f_3$), partially separable functions($f_4 - f_{18}$), fully non-separable functions($f_{19} - f_{20}$) of the CEC'2010 LSGO benchmark functions. CC-GDG-PSO indicates the GDG grouping method is adopted. CC-DGDG-PSO3, CC-DGDG-PSO6, CC-DGDG-PSO9, respectively indicate the DGDG grouping method with different α value $(3, 6, 9)$ is adopted.

Overall, it is seen that CC-DGDG-PSO6 performs much better than CC-DGDG-PSO3 and CC-DGDG-PSO9 over 20 benchmark functions from all tables. The three algorithm obtained the same magnitude global optimal solution over most of functions respectively, while CC-DGDG-PSO6 has a rather small global optimal solution over the three functions f_7, f_8, and f_{12} belonged to partially separable functions, which is orders of magnitude lower than CC-DGDG-PSO3 and CC-DGDG-PSO9.

From Table 1 and Table 3, which respectively consist of fully separable functions and fully non-separable functions, it is obviously that CC-DGDG-PSO performs much better than CC-GDG-PSO algorithm. CC-GDG-PSO performs better than CC-DGDG-PSO only on functions $f_5, f_6, f_{10}, f_{11}, f_{15}, f_{16}$ and f_{18}, from Table 2. But the global optimal solutions they obtained are the same in magnitude. However, CC-DGDG-PSO performs much better than CC-GDG-PSO in most of this functions. This may be caused by the fact that the CC-GDG-PSO is grouped before the algorithm runs and the algorithm reaches convergence early on some functions e.g. f_7, f_8 . However, the CC-DGDG-PSO will be regrouped during the operation of the algorithm so that the originally converged state diverge again to get better results.

Table 1: Minimal, median, mean standard deviation of the fitness values obtained by the 25 independent runs of the compared algorithms on fully separable functions of the CEC'2010 benchmark functions($f_1 - f_3$)

Function		CC-fGDG-PSO	CC-DGDG-PSO3	CC-DGDG-PSO6	CC-DGDG-PSO9
f_1	Min	4.27E-01	8.00E-02	**5.30E-02**	1.30E-01
	Median	1.36E+03	1.74E+00	1.82E-01	4.65E-01
	Mean	1.06E+05	4.93E+00	2.43E-01	5.51E-01
	Std	4.72E+05	8.05E+00	2.18E-01	3.66E-01
f_2	Min	7.36E+03	7.44E+03	7.27E+03	**7.12E+03**
	Median	7.85E+03	7.74E+03	7.66E+03	7.54E+03
	Mean	7.86E+03	7.74E+03	7.62E+03	7.56E+03
	Std	2.30E+02	1.52E+02	1.85E+02	2.15E+02

Table 1: Minimal, median, mean standard deviation of the fitness values obtained by the 25 independent runs of the compared algorithms on fully separable functions of the CEC'2010 benchmark functions($f_1 - f_3$)

Function		CC-fGDG-PSO	CC-DGDG-PSO3	CC-DGDG-PSO6	CC-DGDG-PSO9
f_3	Min	1.20E+01	6.01E+00	4.35E+00	**3.63E+00**
	Median	1.30E+01	6.60E+00	5.10E+00	4.14E+00
	Mean	1.30E+01	6.75E+00	5.11E+00	4.09E+00
	Std	8.00E-01	4.50E-01	4.17E-01	2.59E-01

Table 2: Minimal, median, mean standard deviation of the fitness values obtained by the 25 independent runs of the compared algorithms on partially separable functions of the CEC'2010 benchmark functions($f_4 - f_{18}$)

Function		CC-fGDG-PSO	CC-DGDG-PSO3	CC-DGDG-PSO6	CC-DGDG-PSO9
f_4	Min	4.69E+11	**1.72E+11**	2.84E+11	2.53E+11
	Median	1.06E+12	3.44E+11	5.19E+11	6.23E+11
	Mean	1.14E+12	3.78E+11	5.61E+11	6.54E+11
	Std	4.56E+11	1.23E+11	1.94E+11	2.02E+11
f_5	Min	**2.71E+08**	2.85E+08	3.62E+08	3.13E+08
	Median	3.47E+08	5.32E+08	4.56E+08	4.80E+08
	Mean	3.46E+08	5.09E+08	4.71E+08	4.91E+08
	Std	3.57E+07	1.12E+08	7.13E+07	1.13E+08
f_6	Min	**2.43E+06**	3.54E+06	2.73E+06	2.20E+06
	Median	3.47E+06	4.98E+06	4.05E+06	4.35E+06
	Mean	3.58E+06	8.78E+06	5.95E+06	8.04E+06
	Std	1.00E+06	6.66E+06	5.17E+06	6.42E+06
f_7	Min	5.73E+03	4.40E-01	**9.01E-04**	3.80E-03
	Median	5.95E+05	3.26E+00	2.92E-02	1.02E-01
	Mean	1.21E+05	5.60E+00	8.48E-02	2.11E-01
	Std	1.61E+05	6.79E+00	1.96E-01	2.58E-01
f_8	Min	1.63E+02	1.04E+02	**5.26E-04**	3.01E+02
	Median	3.43E+07	1.15E+07	1.10E-02	2.08E+06
	Mean	4.63E+07	1.53E+07	9.50E+03	1.77E+07
	Std	4.02E+07	2.04E+07	4.70E+04	3.36E+07
f_9	Min	4.97E+06	5.61E+06	4.70E+06	**4.33E+06**
	Median	7.84E+06	7.52E+06	6.11E+06	5.78E+06
	Mean	7.86E+06	7.37E+06	6.23E+06	5.89E+06
	Std	1.31E+06	9.96E+05	7.79E+05	8.96E+05
f_{10}	Min	**7.44E+03**	8.95E+03	8.31E+03	8.50E+03
	Median	7.81E+03	9.73E+03	9.40E+03	9.43E+03
	Mean	7.81E+03	9.82E+03	9.50E+03	9.44E+03
	Std	1.94E+02	5.05E+02	4.93E+02	4.50E+02

Table 2: Minimal, median, mean standard deviation of the fitness values obtained by the 25 independent runs of the compared algorithms on partially separable functions of the CEC'2010 benchmark functions($f_4 - f_{18}$)

Function		CC-fGDG-PSO	CC-DGDG-PSO3	CC-DGDG-PSO6	CC-DGDG-PSO9
f_{11}	Min	**7.70E+01**	1.11E+02	1.02E+02	1.07E+02
	Median	8.60E+01	1.32E+02	1.20E+02	1.24E+02
	Mean	8.60E+01	1.34E+02	1.21E+02	1.25E+02
	Std	4.62E+00	1.70E+01	8.61E+00	8.45E+00
f_{12}	Min	4.60E+01	1.00E+01	9.80E-01	**4.43E-01**
	Median	4.57E+02	1.80E+01	1.83E+00	9.36E-01
	Mean	9.22E+02	2.10E+01	2.00E+00	9.15E-01
	Std	1.47E+03	1.20E+01	7.70E-01	2.41E-01
f_{13}	Min	5.04E+02	3.96E+02	**1.15E+02**	1.50E+03
	Median	9.61E+02	1.06E+03	1.91E+03	2.86E+03
	Mean	1.01E+03	1.22E+03	2.05E+03	3.03E+03
	Std	3.99E+02	5.25E+02	8.25E+02	1.38E+03
f_{14}	Min	1.55E+07	1.61E+07	1.59E+07	**1.30E+07**
	Median	1.81E+07	2.08E+07	1.82E+07	1.70E+07
	Mean	1.83E+07	2.12E+07	1.89E+07	1.66E+07
	Std	1.78E+06	2.48E+06	2.26E+06	1.67E+06
f_{15}	Min	**7.30E+03**	1.03E+04	9.92E+03	9.96E+03
	Median	7.80E+03	1.09E+04	1.08E+04	1.11E+04
	Mean	7.76E+03	1.10E+04	1.09E+04	1.10E+04
	Std	2.67E+02	4.92E+02	6.00E+02	5.10E+02
f_{16}	Min	**1.34E+02**	2.44E+02	2.17E+02	2.18E+02
	Median	1.44E+02	2.57E+02	2.42E+02	2.31E+02
	Mean	1.45E+02	2.60E+02	2.42E+02	2.33E+02
	Std	5.78E+00	1.10E+01	1.00E+01	1.40E+01
f_{17}	Min	1.18E+03	3.18E+02	**2.10E+02**	2.85E+02
	Median	2.51E+03	4.36E+02	3.23E+02	4.19E+02
	Mean	2.73E+03	4.52E+02	3.27E+02	4.22E+02
	Std	1.26E+03	1.04E+02	7.30E+01	7.70E+01
f_{18}	Min	**1.32E+03**	1.71E+03	4.70E+03	3.00E+03
	Median	2.23E+03	3.76E+03	1.05E+04	7.42E+03
	Mean	2.47E+03	4.84E+03	1.12E+04	9.42E+03
	Std	9.17E+02	2.51E+03	4.83E+03	6.32E+03

Table 3: Minimal, median, mean standard deviation of the fitness values obtained by the 25 independent runs of the compared algorithms on fully non separable functions of the CEC'2010 benchmark functions($f_{19} - f_{20}$)

Function		CC-fGDG-PSO	CC-DGDG-PSO3	CC-DGDG-PSO6	CC-DGDG-PSO9
f_{19}	Min	3.28E+05	8.31E+04	**7.63E+04**	8.31E+04
	Median	4.95E+05	1.14E+05	1.14E+05	1.08E+05
	Mean	4.85E+05	1.20E+05	1.13E+05	1.11E+05
	Std	8.61E+04	2.18E+04	2.15E+04	2.13E+04
f_{20}	Min	5.09E+07	2.02E+03	2.13E+03	**1.89E+03**
	Median	2.63E+08	2.43E+03	2.30E+03	2.22E+03
	Mean	3.19E+08	2.43E+03	2.35E+03	2.22E+03
	Std	2.01E+08	1.83E+02	1.68E+02	2.56E+02

5 Conclusions

This paper proposed a Dynamic-GDG method for variable decomposition, which addresses the issue that the GDG method can not be dynamically adjusted with the evolution of the algorithm in the variable decomposition of LSBO problems. By solving 20 test functions of CEC'2010 LSGO benchmark functions, it is known that the optimization result of the algorithm can be improved by the DGDG method. Next, we will continue to study the strategy of dynamic variable decomposition to further improve the optimal effect of LSBO problems.

Acknowledgement. This work was partially supported by the National Natural Science foundation of China (Grant Nos.61673194, 61105128), Key Research and Development Program of Jiangsu Province, China (Grant No.BE2017630), the Postdoctoral Science Foundation of China (Grant No. 2014M560390), Six Talent Peaks Project of Jiangsu Province (Grant No. DZXX-025).

References

1. S. Mahdavi, M. E. Shiri, S. Rahnamayan. Metaheuristics in large-scale global continues optimization: A survey[J]. Information Sciences. 2015, 295: 407-428.
2. Z.-H. Zhou, N. V. Chawla, Y. Jin, G. J. Williams. Big Data Opportunities and Challenges: Discussions from Data Analytics Perspectives [Discussion Forum][J]. IEEE Computational Intelligence Magazine. 2014, 9(4): 62-74.
3. K. Scheerlinck, H. Vernieuwe, B. D. Baets. Zadeh's Extension Principle for Continuous Functions of Non-Interactive Variables: A Parallel Optimization Approach[J]. IEEE Transactions on Fuzzy Systems. 2012, 20(1): 96-108.
4. T. Schaul. Studies in Continuous Black-box Optimization [D]: Universit?t Mnchen, 2011.
5. Y. Mei, M. N. Omidvar, X. Li, X. Yao. A Competitive Divide-and-Conquer Algorithm for Unconstrained Large-Scale Black-Box Optimization[J]. ACM Transactions on Mathematical Software. To Appear.

6. A. Mucherino, M. Fuchs, X. Vasseur, S. Gratton. Variable neighborhood search for robust optimization and applications to aerodynamics. In: Large-Scale Scientific Computing: Springer; 2011:230-237.
7. G. N. Vanderplaats. Very large scale optimization[M]: National Aeronautics and Space Administration, Langley Research Center, 2002.
8. M. A. Potter, K. A. De Jong. A cooperative coevolutionary approach to function optimization. In: Parallel Problem Solving from NaturePPSN III: Springer; 1994:249-257.
9. M. A. Potter, K. A. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents[J]. Evolutionary computation. 2000, 8(1): 1-29.
10. F. van den Bergh, A. P. Engelbrecht. A cooperative approach to particle swarm optimization[J]. IEEE Transactions on Evolutionary Computation. 2004, 8(3): 225-239.
11. Z. Yang, K. Tang, X. Yao. Differential evolution for high-dimensional function optimization[C]. IEEE Congress on Evolutionary Computation: IEEE, 2007,3523-3530.
12. Z. Yang, K. Tang, X. Yao. Large scale evolutionary optimization using cooperative coevolution[J]. Information Sciences. 2008, 178(15): 2985-2999.
13. M. N. Omidvar, X. Li, Z. Yang, X. Yao. Cooperative co-evolution for large scale optimization through more frequent random grouping[C]. IEEE Congress on Evolutionary Computation: IEEE, 2010,1-8.
14. Z. Yang, K. Tang, X. Yao. Multilevel cooperative coevolution for large scale optimization[C]. IEEE Congress on Evolutionary Computation: IEEE, 2008,1663-1670.
15. Z. Yang, Nature Inspired Real-Valued Optimization and Applications [D], University of Science and Technology of China, 2010.
16. R. T., X. Yao. A cooperative coevolutionary algorithm with Correlation based Adaptive Variable Partitioning[C]. IEEE Congress on Evolutionary Computation, 2009,983-989.
17. H. K. Singh, T. Ray. Divide and Conquer in Coevolution: A Difficult Balancing Act. In: Sarker R, Ray T, eds. Agent-Based Evolutionary Search: Springer Berlin Heidelberg; 2010:117-138.
18. M. N. Omidvar, Y. Mei, X. Li. Effective decomposition of large-scale separable continuous functions for cooperative co-evolutionary algorithms[C]. IEEE Congress on Evolutionary Computation, 2014,1305-1312.
19. J. Liu, K. Tang. Scaling Up Covariance Matrix Adaptation Evolution Strategy Using Cooperative Coevolution. In: Yin H, Tang K, Gao Y, et al., eds. Intelligent Data Engineering and Automated Learning C IDEAL 2013: Springer Berlin Heidelberg; 2013:350-357.
20. J. Liu. CMA–ES and Decomposition Strategy for Large Scale Continuous Optimization Problem[D]: University of Science and Technology of China, 2014.
21. M. N. Omidvar, X. Li, Y. Mei, X. Yao. Cooperative co-evolution with differential grouping for large scale optimization[J]. IEEE Transactions on Evolutionary Computation. 2014, 18(3): 378-392.
22. K. Tang, X. Li, P. N. Suganthan, Z. Yang, T. Weise. Benchmark Functions for the CEC2010 Special Session and Competition on Large-Scale Global Optimization; 2010.
23. M. N. Omidvar, X. Li, K. Tang. Designing benchmark problems for large-scale continuous optimization[J]. Information Sciences. 2015, 316: 419-436.