Journal of Computers Vol. 29 No. 5, 2018, pp. 129-141
doi:10.3966/199115992018102905011

Cooperative Co-evolution Algorithm with Problem Adaptive Variable Grouping for Large Scale Global Optimization

Fei Wei[1*], Shugang Li[2], Jinfeng Xue[3]

[1] College of Sciences, Xi'an University of Science and Technology, Xi'an, 710054, China
feiweixjf@163.com

[2] College of Safety Science and Engineering, Xi'an University of Science and Technology, Xi'an, 710054, China

[3] 31662 Army, Lin xia, 731100, China

Received 7 June 2017; Revised 7 November 2017; Accepted 7 December 2017

Abstract. For large scale global optimization problems, evolutionary algorithms (briefly, EAs) will face a huge challenge, and their efficiency and effectiveness will be much reduced. To enhance their efficiency and effectiveness, in this paper, a problem adaptive variable grouping strategy (briefly, PAVG) is firstly proposed. In PAVG, we make the grouping directly via the expression of the objective function which usually consists of finite number of operations of four arithmetic operations " $+$, $-$, $f2$, $\div$ " and composite operation of basic elementary functions. We classify these operations into two classes: one will result in non-separable variables, and the other will result in separable variables. In this way, the variables can be grouped into several non-interacting subcomponents, while the variables in each subcomponent are interactive. Then, combining with PAVG, a novel algorithm, called cooperative co-evolution algorithm with problem adaptive variable grouping strategy (briefly, CCPA), is designed, and evolution can be conducted in these subcomponents separately. In this way, a large-scale problem can be decomposed into several small-scale problems and this makes the problem solving much easier. To further enhance the efficiency of CCPA, a new local search scheme is designed, and when a good solution is found in the evolution process (e.g., by crossover and mutation), it will be further improved by the local search scheme. To verify the efficiency of CCPA, the simulations are made on the standard benchmark suites of CEC'2010 and CEC'2013, and CCPA is compared with several well performed algorithms. The results indicate that the proposed algorithm CCPA is more efficient and effective.

Keywords: cooperative co-evolution, initialization method, large scale global optimization, problem adaptive, variable grouping strategy

1 Introduction

Many real-world optimization problems involve a large number of decision variables. For example, in shape optimization, a large number of shape design variables are often used to represent complex shapes, such as turbine blades, aircraft wings, and heat exchangers, etc. How to handle this sort of real-world large scale global optimization (LSGO) problems efficiently still remains an open problem.

In recent years, many new theoretical and computational contributions have been reported for solving LSGO problems, e.g., cooperative co-evolution (briefly, CC) [1-11], decomposition methods [12-13], estimation of distribution algorithms [14-15], memetic algorithms [16-17], etc. Moreover, to test the effectiveness of the large scale global optimization algorithms, some benchmark suites have been proposed (e.g., [18-19]).

* Corresponding Author

For LSGO problems, a natural approach is to adopt a divide-and-conquer strategy. In existing approaches, CC is proposed by Potter and De Jong [1], which is one of the most efficient and popular algorithms. First, it divides a problem into several smaller subcomponents, then each subcomponent is evolved by using a separate evolutionary algorithm (EA) [1]. In the early stage of the development of CC, two decomposition methods: one-dimension based and splitting-in-half methods were adopted. These two methods do not take the interaction between variables into consideration. Thus, they cannot solve problems consisting of non-trivial variable interactions efficiently. In order to mitigate this problem, recently, some effective grouping methods (e.g., [2-8]) have been successively proposed. For example, Yang *et al*. proposed a random grouping method (used in DECC-G [3], MLCC [4]). However, the random grouping method just increases the probability of two interacting variables allocated into the same subcomponent; Chen *et al.* proposed a variable interaction learning grouping strategy in CCVIL [5] which takes the advantage of variable correlation information. However, it is only a sufficient condition rather than a sufficient and necessary condition to find interacting variables. More importantly, the number of problem evaluations needed in CCVIL is very large for LSGO. Recently, Mohammad and Li et al. proposed an automatic decomposition strategy called differential grouping (DECC-DG) [7] that is a great improvement to the existing grouping strategies. The experiments on CEC'2010 benchmark suite [18] have shown this grouping strategy is very efficient. However, the parameter ε involved in the differential grouping will affect the effect of the grouping, and the condition given in the differential grouping is only a sufficient condition rather than a sufficient and necessary condition to find interacting variables. In addition, some of the aforementioned algorithms just tested the small scale benchmark suites (e.g., [20-21]). To overcome aforementioned shortcomings, in this paper, we propose a novel problem adaptive variable grouping strategy based on the formulation characteristic of a general function. In this variable grouping strategy, we make the grouping directly via the expression of the objective function. Note that a general function consists of finite number of operations of four arithmetic operations "$+,-,\times,\div$" and composite operation of basic elementary functions. We classify these operations into two classes: one will result in the variables non-separable, and the other will result in the variables separable. In this way, we do not need to determine any parameter and the strategy can be applied any function.

Except for the novel grouping strategy, to further improve the efficiency and effectiveness of CC, a local search strategy is designed and integrated into CC, and when a good solution is found in the evolving process (e.g., by crossover and mutation), it will be further improved by the local search. Note that there have been some very efficient local search algorithms, e.g., Quasi Newton algorithm and conjugate gradient algorithm. However, these algorithms require the gradients of the function, and cannot be applicable to non-differentiable problems. To keep the efficiency of these local search algorithms and avoid computing the gradients, a revised version of Quasi Newton algorithm is designed and as the local search scheme in this paper.

Based on the above ideas, the goal of the paper is to develop a new cooperative coevolution algorithm with problem adaptive variable grouping strategy (briefly, CCPA), so that CCPA has the following advantages: (1) it can decompose a large scale problem into several small scale problems if variables are divisible; (2) it can search for multiple areas in the search space simultaneously (search for each decomposed problem respectively); (3) it can find a better local optimal solution (via the local search scheme). The simulations are made on two benchmark suites (CEC'2010 [18] and CEC'2013 [19]), and CCPA is compared with several efficient algorithms.

The reminder of the paper is organized as follows. In section 2, first, an initialization method is introduced; second, a variable grouping strategy is proposed respectively; third, a local search strategy is designed; finally, a novel algorithm framework CCPA is presented. Numerical experiments are given in Section 3. Section 4 presents conclusions and future works. In this paper, we adopt the following notations:

k: the generation number;

D: the dimension of the test problems;

x_k^*: the local minimizer of the objective function in the k-th generation;

f_k^*: the function value at x_k^*;

x^*: the global minimizer of the objective function;

FEs: the number of fitness evaluations;

MaxFEs: the maximum number of fitness evaluations.

N-Sep: a set of basic elementary functions and four arithmetic operations affecting variables interaction.

2 Cooperative Co-evolution

The basic idea of CC is to adopt divide-and-conquer strategy for a problem, and the critical steps of CC can be summarized as follows:

(1) Problem decomposition: decompose a high-dimensional vector into smaller subcomponents that can be handled by conventional EAs.

(2) Subcomponent optimization: evolve each subcomponent separately using a certain EA.

(3) Subcomponent combination: merge solutions of all subcomponents, which constitute a solution of the original problem.

For CC, like other population-based optimization algorithms, the initial population is an important issue. In the following, we introduce an initialization method.

2.1 Initialization Method

Traditionally, basic random number generators (RNGs) are widely used to initialize the population of CC. Recently, a large and growing body of literatures has proposed new ways to generate initial population (e.g., [9-11]). In this paper, we select the chaotic method [10] as a population generator. The detailed process of chaos initialization is as follows:

Using the following formula to generate the chaos factor:

$$z_j^{(i+1)} = \mu(1-2\mid z_j^{(i)} - 0.5\mid), 0 \le z_j^{(0)} \le 1, j = 1,2,\cdots,D. \quad \textbf{(1)}$$

where z_j denotes the *j-th* chaos factor, and i denotes the chaos iteration number. Set $i=0$ first and randomly generate D uniform factors, and in this paper, we use $\mu = 1$. Then, let $i = 1,2,\cdots,m$ successively and generate the initial swarms.

After that, these chaos factors $z_j^{(i)}, i = 1,2,\cdots,m$ will be mapped into the search space of the decision variables as follows:

$$x_{ij} = x_{\min,j} + z_j^{(i)}(x_{\max,j} - x_{\min,j}), j = 1,2,\cdots,D. \quad \textbf{(2)}$$

and

$$X_i = (x_{i1}, x_{i2}, \cdots, x_{iD}), i = 1,2,\cdots,m. \quad \textbf{(3)}$$

are the initial chaos swarms.

2.2 Problem Adaptive Variable Grouping Strategy: PAVG

The main idea of CC is the decomposition of problem into smaller subcomponents, then every subcomponent is evolved by an evolutionary algorithm (EA) separately. The ideal goal is that the decision variables with interaction are grouped into one subcomponent, and the decision variables in any two subcomponents are not interacted. In this way, a large-scale problem can be divided into separate subproblems with lower dimensions. However, precisely grouping is a hard task.

At present, some efficient grouping strategies have been proposed, such as randomly grouping (DECC-G [3], MLCC [4]), variable interaction learning grouping (CCVIL [5]), route distance grouping [6], differential grouping (DECC-DG) [7], delta grouping (DECC-DML) [8], and so on. However, these strategies still cannot accurately realize grouping.

To overcome this shortcoming, in this paper, we propose a new problem adaptive variable grouping method (briefly, PAVG) that can divide all variables into several subcomponents accurately and self-adaptively.

The PAVG framework is as follows: First, we classify the situations of variable non-separability and

build a set *N-Sep* whose elements are those operations (among arithmetic operations and composite operations) resulting in variables interaction; second, searching the variables involved in the operations in *N-Sep* in the expression of objective function; finally, group the variables involved in any operation in *N-Sep* into a subcomponent.

Note that an expression of the objective function in a general optimization problem consists of the finite number of four arithmetic operations "$+,-,\times,\div$" and composite operation of basic elementary functions (i.e., power function y^a, exponential functions a^y and e^y, logarithmic functions $\ln y$ and $\log_a y$, trigonometric functions $\sin y$, $\cos y$, $\tan y$, $\cot y$, $\sec y$ and $\arcsin y$, inverse trigonometric functions $\arcsin y$, $\arccos y$, $\arctan y$, $\operatorname{arccot} y$, $\operatorname{arcsec} y$ and $\operatorname{arccsc} y$, and constant, where $y \in R$). We classify the situations of variables non-separability based on the exp ression structure of objective functions according to the following cases;

(1) Variables non-separability in four arithmetic operations. If function $p(x) = a_1x_1 + a_2x_2 + \cdots + a_mx_m$, then each a_ix_i in this function can be optimized independently and thus the variable $x_1, x_2, \cdots, x_m$ in this function are separable, where $a_i \geq 0$, $i = 1,2,\cdots,m$. If $a_i \in R$ for $i = 1,2,\cdots,m$ in function $p(x) = a_1x_1 + a_2x_2 + \cdots + a_mx_m$, then we can write $p(x)$ into another equivalent form $p(x) = b_1x_1 + b_2x_2 + \cdots + b_mx_k - (c_{k+1}x_{k+1} + \cdots + c_mx_m)$, where b_i and c_i are positive for $i = 1 \sim k$ and $j = k+1 \sim m$. Thus, $x_1, x_2, \cdots, x_k$ are separable and $x_{k+1}, \cdots, x_m$ are separable. As a result, variables in $p(x)$ are separable. While if a function contains "$\times$" or "$\div$" of two variables, these two variables cannot be optimized independently and thus they are non-separable. We put two operations "$\times$" and "$\div$" into set $N - Sep$.

(2) Variables non-separability in a composite function. For a basic elementary function $\mathrm{g}(\mathrm{y})$ with $y \in R$ and an n-dimensional function $h(x)$, if $g(y)$ is monotone and variables in function $h(x)$ are separable, then variables in composite function $g(h(x))$ are also separable (e.g., if $g(y) = e^y$, and $h(x) = x_1 + x_2 + \cdots + x_{10}$, then variables in $g(h(x)) = e^{x_1 + x_2 + \cdots + x_{10}}$ are separable). Otherwise, variables in $g(h(x))$ are non-separable, i.e., when $g(y)$ is a non-monotonic function or $h(x)$ is non-separable ($h(x)$ can be seen as a composite function of basic elementary functions and other functions), their composite function $g(h(x))$ is non-separable. Thus, the key operation to form a non-separable composite function for this case is that $g(y)$ is non-monotone, and we put the non-monotonic basic elementary functions (e.g., trigonometric functions, inverse trigonometric functions, and power function with $y \in R$) into set *N-Sep*.

(3) Variables non-separability in a function obtained by one operation of "$+,-,\times,\div$" on two composite functions in (2). Note that operations "$+$" and "$-$" do not change the variables separability. The variables in "$\times$" or "$\div$" of two composite functions in (2) are non-separable except for both the composite functions are exponential functions. Thus, we put "$\times$" and "$\div$" into set *N-Sep* (except that both the composite functions are exponential functions).

Through above three cases, we can determine the groups of interactive variables and obtain the set *N-Sep* whose elements (in fact, some operations) link the interactive variables. Through set *N-Sep*, we can design the following algorithm PAVG to group the variables into different subcomponents.

The framework of the variable grouping strategy PAVG.

(1) For a given problem, construct set *N-Sep* according to above three cases, then the elements in *N-Sep* can be seen as some strings (such as "$\cos$", "$\sin$" and "$\times$", etc.);

(2) If some variables are linked by an element in *N-Sep*, then put these variables into a subcomponent. If other variables are also linked to the variables in this subcomponent by an element in *N-Sep*, then these additional variables are also put into this subcomponent. Note that if we have checked the non-separability of some variables in a function in previous steps, when we meet this function again in the objective function, we can directly use this result in the following steps instead of checking the term by term of this function again. This will improve the efficiency of grouping. For example, if we have checked variables x_1 and x_5 in $\cos(x_1x_5)$ are non-separable, when we meet $\cos(x_ix_j)$ in the following check, we can directly get the result that x_i and x_j are non-separable instead of making the check process

from the beginning.

(3) Repeat step 2 until all variables and terms in the expression of objective function are checked. As a result, some variables will be grouped into some subcomponents (subcomponent), and the remaining variable(s) will not be grouped.

(4) The variables which are not grouped in step 3 are separable, and each of these variables is grouped into a subcomponent.

In this section, an example is given to illustrate the variable grouping strategy PAVG.

For example: $f(x)=x_1\times x_2+\sin(x_1+x_3)-\tan(x_4+x_5)+\exp(x_5)\times\exp(x_6)$. We know that $\{\times,\sin\}$ is contained in *N-Sep* and we match these strings with *N-Sep* by using regular expressions. Then we can obtain that x_1 and x_2 are related, x_1 and x_3 are related. Trigonometric function *tan*(x) is monotone, so x_4 and x_5 are not related. Though $\exp(x_5)\times\exp(x_6)$ contain " × ", $\exp(x_5)\times\exp(x_6)=\exp(x_5+x_6)$ and exponential function $\exp(x)$ is monotone, so x_5 and x_6 are not related. Therefore, x_1 , x_2 and x_3 are related and put them in a subcomponent, and x_4 , x_5 and x_6 are separately put into different subcomponents. Thus, all variables are grouped into four subcomponents.

2.3 A New Local Search Strategy

To further enhance the efficiency of the proposed algorithm CCPA which will be described in Section 3, we design a new local search scheme.

When a good solution is found in the evolving process (e.g., by crossover and mutation) of an EA, it may not be a local optimal solution, and it usually can be further improved by local search. Thus, after evolving process in each generation, a local search strategy would be helpful to improve the searching efficiency of the algorithm. There have been some efficient local search algorithms, e.g., Conjugate Gradient Method, Newton Method and Quasi Newton Method, etc. However, these local search algorithms require the gradients of the function, and are not suitable for solving non-differentiable problems. To keep the advantages of these local search algorithms and avoid computing the gradients, a revised version of Quasi Newton algorithm is designed and its pseudocode is shown in algorithm 1.

Algorithm 1. Pseudocode of the local search strategy

1 **Initialization:** Choose a tolerance $\varepsilon<0$, e.g. $\varepsilon=1.0\text{e-}5$, and a small constant $\delta\in(0,1),\sigma\in(0,0.5),$ and $\Delta x=1.0e\text{-}1$.

2 Give an initial point χ_0 and an approximate inverse of the Hessian matrix B_0 ,

3 $k=0.$

4 Calculate the approximate *i-th* component of gradient vector g_k by $g_{ki}=(f(x_k+\Delta xe_i)-f(x_k))/\Delta x$ for $i=1+n,$ where $e_i=(0,\cdots,0,1,0,\cdots,0)^T$ with the *i-th* component of e_i being one.

5 **Repeat**

6 Obtain a direction d_k *by* $d_k=-B_kg_k$.

7 Perform a line search based on the Armijo criterion [25] to find an acceptable step size $\lambda_k=\delta^{mk}$, where m_k is the smallest non-negative integer that satisfy the following inequality:

$$f(x_k+\delta^{mk}d_k)\le f(x_k)+\sigma\delta^{mk}g_k^Td_k\ .$$

8 Set $s_k=\lambda_kd_k$, and $x_{k+1}=x_k+s_k$, then calculate the approximate *i*-th component of gradient vector g_{k+1} by $g_{k+1}=\left(f\left(x_{k+1}+\Delta\mathrm{x}e_i\right)-f\left(x_k+1\right)\right)/\Delta x;$

for $i=1\sim n$ and set $\mathrm{y}_k=g_{k+1}-g_k$, then

$$\begin{cases}B_{k+1}=B_k+(\beta_ks_ks_k^T-B_ky_ks_k^T-s_ky_k^TB_k)/\ s_k^Ty_k,\\ \beta_k=1+y_k^TB_ky_k/\ s_k^Ty_k.\end{cases}$$

9 Let $k=k+1$,

10 **untill** $\|g_k\|\le\varepsilon$;

11 $x^*=x_k$,

12 **return** x^*.

2.4 Cooperative Co-evolution Algorithm with Problem Adaptive Variable Grouping Strategy (CCPA)

For large scale optimization problems, we have proposed a variable grouping method in algorithm PAVG. Using this algorithm, a large-scale optimization problem can be decomposed into several small-scale optimization problems. For these small-scale optimization problems, it is important to use an efficient algorithm to solve them in order to further enhance the performance of the whole algorithm.

It is well known that Differential Evolution (DE) is very effective and simple [22], but the general DE suffers the sensitivity of its control parameters [23]. It has been indicated that self-adaptive differential evolution with neighborhood Search (SaNSDE) [24] performs significantly better than other similar DE algorithms due to its self-adapted crossover rate *CR* and scaling factor *F* [24], and SaNSDE has been successfully applied in a variety of problems (e.g., [3-4]). Thus, we choose this algorithm as the evolutionary algorithm in each subcomponent. Based on algorithm 1 and SaNSDE, we design a new algorithm: a cooperative co-evolution algorithm with problem adaptive variable grouping strategy (CCPA). Its pseudocode is given in Algorithm 2.

Algorithm 2. Pseudocode of CCPA

Initialization: let $K = 0, FEs = 0$ and ε is a tolerance threshold, i is the iteration number. Generate initial population POP, where $|POP| = N$;
Perform PAVG, and obtain $SubPOP(j), j = \{1, 2, \cdots, M\}$;
repeat
  for $j = 1, \cdots, M$ do
    while $f(x_i) - f(x_{i+1}) > \varepsilon$ do
    optimize $SubPOP(j)$ by SaNSDE;
  end
  record the current best solution. y_j^* corresponding to $SubPOP(j)$; then update FEs.
end
Update current best solution by $x_{K+1}^* = Best\{x_K^*, y_1^*, \cdots, y_M^*\}$,
Generate $N - (M+1)$ new individuals, and use these individuals and $x_K^*, y_1^*, \cdots, y_M^*$ as the new initial population; $K = K + 1$;
until $MaxFES$ is met;
$x^* = x_K^*$ and $f(x^*) = f(x_K^*)$;
return x^* and $f(x^*)$.

It's worth mentioning that. after the variables grouping, SaNSDE can be used to optimize each subcomponent, respectively; however, for a very small subcomponent (e.g., the size of subcomponent is equal to one), using SaNSDE may waste *FEs*. In order to reduce the cost of *FEs* , an efficient deterministic method (e.g., algorithm 1) can be used to solve the very small subcomponent problem.

3 Numerical Experiments

3.1 Benchmark Suite and Parameters Setting for CCPA

In this section, first, *CEC'2010* benchmark suite [18] is tested, where the dimensions of the problems are 1000, and the results are recorded in Table 1; second, *CEC'2013* benchmark suite [19] is tested, and the results are recorded in Table 2, where $f1 - f3$ are fully separable functions, and $f4 - f11$ are partially additively separable functions, and $f12 - f15$ are non-separable functions, and the dimensions of the problems are 1000.

Table 1. Comparison between CCPA and other algorithms on CEC'2010 benchmark suite, D=1000

P		CCPA	MA-SW-Chains	DECC-DG	CCVIL	DECC-G	MLCC	DECC-DML
f1	mean	**0.00E+00**	2.01e-14	5.47e+03	1.55e-17	2.93e-07	1.53e-27	1.93e-25
	std	**0.00E+00**	1.99e-14	2.02e+04	8.62e-08	7.75e-17	7.66e-27	1.86e-25
f2	mean	5.16E+01	8.10e+02	4.39e+03	**6.71e-09**	1.31e+03	5.57e-01	2.17e+02
	std	1.42E+01	5.88e+01	1.97e+02	3.24e+01	**2.31e-08**	2.21e+00	2.98e+01
f3	mean	1.10E-13	7.28e-13	1.67e+01	7.52e-11	1.39e+00	9.88e-13	**1.18e-13**
	std	1.56E-15	3.40e-13	3.34e-01	9.59e-02	6.58e-11	3.70e-12	**8.22e-15**
f4	mean	**4.84E+10**	3.53e+11	4.79e+12	9.62e+12	5.00e+12	9.61e+12	3.58e+12
	std	**2.28E+10**	3.12e+10	1.44e+12	3.38e+12	3.43e+12	3.43e+12	1.54e+12
f5	mean	**8.08E+07**	1.68e+08	1.55e+08	1.76e+08	2.63e+08	3.84e+08	2.98e+08
	std	**1.71E+07**	1.04e+08	2.17e+07	8.44e+07	6.47e+07	6.93e+07	9.31e+07
f6	mean	3.42E+06	8.14e+04	**1.64e+01**	2.94e+05	4.96e+06	1.62e+07	7.93e+05
	std	1.36E+07	2.84e+05	**2.71e-01**	8.02e+05	6.09e+05	4.97e+06	3.97e+06
f7	mean	**2.03E-10**	1.03e+02	1.16e+04	8.00e+08	1.63e+08	6.89e+05	1.39e+08
	std	**2.27E-12**	8.70e+01	7.41e+03	1.38e+08	2.48e+09	7.37e+05	7.72e+07
f8	mean	**1.28E+06**	1.41e+07	3.04e+07	6.50e+07	6.44e+07	4.38e+07	3.46e+07
	std	**1.90E+06**	3.68e+07	2.11e+07	2.89e+07	3.07e+07	3.45e+07	3.56e+07
f9	mean	**7.65E+06**	1.41e+07	5.96e+07	6.66e+07	3.21e+08	1.23e+08	5.92e+07
	std	**9.55E+05**	1.15e+06	8.18e+06	3.39e+07	1.60e+07	1.33e+07	4.71e+06
f10	mean	1.31E+04	**2.07e+03**	4.52e+03	1.28e+03	1.06e+04	3.43e+03	1.25e+04
	std	3.47E+02	1.44e+02	1.41e+02	2.93e+02	**7.95e+01**	8.72e+02	2.66e+02
f11	mean	2.56E+01	3.80e+01	1.03e+01	3.48e+00	2.34e+01	1.98e+02	**1.80e-13**
	std	2.50E+00	7.35e+00	1.01e+00	1.79e+00	1.91e+00	6.98e-01	**9.88e-15**
f12	mean	7.20E-03	**3.62e-06**	2.52e+03	8.95e+03	8.93e+04	3.49e+04	3.79e+06
	std	2.49E-02	**5.92e-07**	4.86e+02	6.90e+03	5.39e+03	4.92e+03	1.50e+05
f13	mean	**5.58E+01**	1.25e+03	4.54e+06	5.72e+02	5.12e+03	2.08e+03	1.14e+03
	std	**4.43E+01**	5.72e+02	2.13e+06	3.95e+03	2.55e+02	7.27e+02	4.31e+02
f14	mean	5.54E+07	**3.11e+07**	3.41e+08	1.74e+08	8.08e+08	3.16e+08	1.89e+08
	std	4.79E+06	**1.93e+06**	2.41e+07	6.06e+07	2.68e+07	2.77e+07	1.49e+07
f15	mean	4.32E+03	2.74e+03	5.88e+03	**2.65e+03**	1.22e+04	7.11e+03	1.54e+04
	std	1.28E+02	1.22e+02	1.03e+02	9.10e+02	**9.34e+01**	1.34e+03	3.59e+02
f16	mean	1.92E+01	9.98e+01	**7.39e-13**	7.18e+00	7.66e+01	3.76e+02	5.08e-02
	std	3.05E+00	1.40e+01	**5.70e-14**	8.14e+00	2.23e+00	4.71e+01	2.54e-01
f17	mean	7.18E+02	**1.24e+00**	4.01e+04	2.13e+04	2.87e+05	1.59e+05	6.54e+06
	std	2.86E+02	**1.25e-01**	2.85e+03	1.97e+04	9.16e+03	1.43e+04	4.63e+05
f18	mean	**1.29E+03**	1.30e+03	1.11e+10	1.33e+04	2.46e+04	7.09e+03	2.47e+03
	std	**1.31E+02**	4.36e+02	2.04e+09	1.05e+04	1.00e+04	4.77e+03	1.18e+03
f19	mean	9.22E+05	**2.85e+05**	1.74e+06	3.52e+05	1.11e+06	1.36e+06	1.59e+07
	std	3.91E+04	**1.78e+04**	9.54e+04	5.00e+04	2.04e+04	7.35e+04	1.72e+06
f20	mean	2.49E+03	1.07e+03	4.87e+07	1.11e+03	4.06e+03	2.05e+03	**9.91e+02**
	std	2.21E+02	7.29e+01	2.27e+07	3.66e+02	3.04e+02	1.80e+02	**3.51e+01**

Table 2. Comparison between CCPA and other algorithms on CEC'2013 benchmark suite, D=1000

P		CCPA	CCR	SACC	MOS	DECC-G
f1	Best	0.00e+00	0.00e+00	0.00e+00	**0.00e+00**	1.75e-13
	Median	0.00e+00	0.00e+00	0.00e+00	**0.00e+00**	2.00e-13
	Worst	0.00e+00	1.81e-03	6.81e-23	**0.00e+00**	2.45e-13
	Mean	0.00e+00	2.58e-04	2.73e-24	**0.00e+00**	2.03e-13
	Std	0.00e+00	6.84e-04	1.36e-23	**0.00e+00**	1.78e-14
f2	Best	1.79e+01	1.91e+02	2.88e+02	**7.40e+02**	9.90e+03
	Median	3.12e+01	1.15e+03	5.71e+02	**8.36e+02**	1.03e+03
	Worst	7.61e+01	4.11e+03	2.72e+03	**9.28e+02**	1.07e+03
	Mean	3.96e+01	2.11e+03	7.06e+02	**8.32e+02**	1.03e+03
	Std	2.20e+01	1.73e+03	4.72e+02	**4.48e+01**	2.26e+01

Table 2. Comparison between CCPA and other algorithms on CEC'2013 benchmark suite, D=1000 (continue)

P		CCPA	CCR	SACC	MOS	DECC-G
f3	Best	3.97e-13	2.06e-13	9.24e-14	**8.20e-13**	2.26e-10
	Median	4.22e-13	2.70e-13	1.21e+00	**9.104e-13**	2.85e-10
	Worst	4.58e-13	3.32e-00	3.76e+00	**1.00e-12**	3.16e-10
	Mean	4.32e-13	3.76e-01	1.11e+00	**9.17e-13**	2.87e-10
	Std	2.18e-14	9.96e-01	1.11e+00	**5.12e-14**	1.38e-11
f4	Best	2.20e+07	7.02e+09	8.48e+09	**1.10e+08**	7.58e+09
	Median	4.47e+08	2.79e+10	3.66e+10	**1.56e+08**	2.12e+10
	Worst	1.14e+09	1.35e+11	1.71e+11	**5.22e+08**	6.99e+10
	Mean	5.83e+08	6.09e+10	4.56e+10	**1.74e+08**	2.60e+10
	Std	3.35e+08	5.46e+10	3.60e+10	**7.87e+08**	1.47e+10
f5	Best	2.60e+08	1.46e+06	3.36e+06	**5.25e+06**	7.28e+14
	Median	2.99e+06	1.02e+07	6.95e+06	**6.79e+06**	7.28e+14
	Worst	4.67e+06	1.37e+07	1.40e+07	**8.56e+08**	7.28e+14
	Mean	3.35e+06	8.31e+06	7.74e+06	**6.94e+06**	7.28e+14
	Std	7.30e+06	4.90e+06	3.22e+06	**8.85e+05**	1.51e+05
f6	Best	1.17e+05	1.21e+05	1.57e+05	**1.95e+01**	6.96e-08
	Median	1.28e+05	1.38e+05	2.07e+05	**1.39e+05**	6.08e+04
	Worst	1.52e+05	1.82e+05	6.00e+05	**2.31e+05**	1.10e+05
	Mean	1.30e+05	1.46e+05	2.47e+05	**1.48e+05**	4.85e+04
	Std	1.25e+04	2.56e+04	1.02e+05	**6.43e+04**	3.98e+04
f7	Best	3.84e+02	2.44e+07	1.72e+06	**3.49e+03**	1.96e+08
	Median	1.90e+03	3.13e+08	1.58e+07	**1.62e+04**	4.27e+08
	Worst	1.09e+03	1.02e+09	1.18e+09	**3.73e+04**	1.78e+09
	Mean	3.08e+03	4.65e+08	8.98e+07	**1.62e+04**	6.07e+08
	Std	3.62e+03	4.22e+08	2.48e+08	**9.10e+03**	4.09e+08
f8	Best	5.53e+14	5.33e+13	1.47e+14	**3.26e+12**	1.43e+14
	Median	1.15e+15	1.57e+15	9.86e+14	**8.18e+12**	3.88e+14
	Worst	2.35e+15	5.62e+15	3.08e+15	**1.32e+13**	7.75e+14
	Mean	1.40e+15	2.14e+15	1.20e+15	**8.00e+12**	4.26e+14
	Std	6.98e+14	1.77e+15	7.63e+14	**3.07e+12**	1.53e+14
f9	Best	2.52e+08	2.54e+08	2.29e+08	2.63e+08	**2.20e+08**
	Median	**3.72e+08**	4.04e+08	5.77e+08	3.87e+08	4.17e+08
	Worst	**3.81e+08**	4.89e+08	1.01e+09	5.42e+08	6.55e+08
	Mean	**3.45e+08**	3.75e+08	5.98e+08	3.83e+08	4.27e+08
	Std	**4.70e+07**	7.97e+07	2.03e+08	6.29e+07	9.89e+07
f10	Best	**1.34e+02**	5.92e+06	1.38e+07	5.92e+02	9.29e+04
	Median	**1.83e+02**	1.09e+07	2.11e+07	1.18e+06	1.19e+07
	Worst	**2.37e+02**	1.59e+07	.75e+07	1.23e+06	1.73e+07
	Mean	**1.91e+02**	1.02e+07	2.95e+07	9.02e+05	1.10e+07
	Std	**4.20e+01**	3.16e+06	1.93e+07	5.07e+05	4.00e+06
f11	Best	8.36e+07	3.35e+08	8.12e+07	**2.06e+07**	4.68e+10
	Median	9.85e+07	1.88e+10	5.30e+08	**4.48e+07**	1.60e+11
	Worst	1.55e+08	2.93e+11	2.30e+10	**9.50e+04**	7.16e+11
	Mean	1.08e+08	1.01e+11	2.78e+09	**5.22e+07**	2.46e+11
	Std	2.79e+07	1.28e+11	5.90e+09	**2.05e+07**	2.03e+11
f12	Best	2.20e+03	2.48e+03	2.43e+02	**2.22e-01**	9.80e+02
	Median	2.51e+03	2.71e+03	8.74e+02	**2.46e+02**	1.03e+03
	Worst	3.59e+03	3.00e+03	1.72e+03	**1.17e+03**	1.20e+03
	Mean	2.59e+02	2.71e+03	8.73e+02	**2.47e+02**	1.04e+03
	Std	4.65e+02	1.81e+02	3.71e+02	2.54e+02	**5.76e+01**
f13	Best	3.54e+08	3.86e+09	6.72e+08	**1.52e+06**	2.09e+10
	Median	7.53e+08	4.78e+09	1.51e+09	**3.30e+06**	3.36e+10
	Worst	3.41e+09	7.45e+09	3.40e+09	**6.16e+06**	4.64e+10
	Mean	1.16e+09	5.21e+09	1.78e+09	**3.40e+06**	3.42e+10
	Std	1.04e+09	1.30e+09	8.05e+08	**1.06e+06**	6.41e+09

Table 2. Comparison between CCPA and other algorithms on CEC'2013 benchmark suite, D=1000 (continue)

P		CCPA	CCR	SACC	MOS	DECC-G
f14	Best	7.22e+09	3.91e+08	8.21e+07	**1.54e+07**	1.91e+11
	Median	1.79e+10	7.96e+09	7.34e+09	**2.42e+07**	6.27e+11
	Worst	3.69e+10	2.52e+11	1.10e+11	**4.46e+07**	1.04e+12
	Mean	1.96e+10	4.75e+10	1.75e+10	**2.56e+07**	6.08e+11
	Std	9.77e+09	9.21e+10	2.87e+10	**7.94e+06**	2.06e+11
f15	Best	2.91e+06	3.91e+08	**1.26e+06**	2.03e+06	4.63e+07
	Median	3.44e+06	6.22e+06	**1.88e+06**	2.38e+06	6.01e+07
	Worst	3.87e+06	7.56e+06	4.90e+06	**2.88e+06**	7.15e+07
	Mean	3.39e+06	5.32e+06	**2.01e+06**	2.35e+06	6.05e+07
	Std	3.48e+05	2.06e+06	7.23e+05	**1.94e+05**	6.45e+06

In experiments, CCPA was tested on an Intel (R) Core (TM) i7 CPU 870 with 2.93GHz in Matlab R2012a.

3.2 The Simulation Results

CCPA is executed 25 independent runs for each test problem using stop criterion *MaxFEs*, and we record the smallest fitness value in 25 runs, denoted as "Best"; the median fitness value in 25 runs, denoted as "Median"; the largest fitness value in 25 runs, denoted as "Worst"; the average fitness value in 25 runs, denoted as "Mean"; and the standard deviation in 25 runs, denoted as "Std". These results are recorded in Tables 1 and 2; and the best result of all compared algorithms are shown in bold.

In Table 1, six algorithms (MA-SW-Chains [26], DECC-DG [7], CCVIL [5], DECC-G [3], MLCC [4], DECC-DML [8]) are compared with CCPA. From Table 1, it can be seen that CCPA is obviously superior to others for 9 test problems. Respectively, CCPA is better than MA-SW-Chains for 12 test problems; CCPA is better than DECC-DG for 16 test problems; CCPA is better than CCVIL for 12 test problems; CCPA is better than DECC-G for 18 test problems; CCPA is better than MLCC for 17 test problems; CCPA is better than DECC-DML for 15 test problems. On the whole, CCPA is obviously superior to the other comparison algorithms.

To show the performance of the proposed CCPA, we compare CCPA with four algorithms (CCR, SACC [27], MOS [28] and DECC-G [3]) on CEC'2013 benchmark suite. In order to test the efficiency of the proposed variable grouping strategy, we replace the proposed variable grouping strategy by random grouping strategy [4] in CCPA and the resulted algorithm is denoted as CCR and the group size was set to 100. The main difference between SACC and CCPA is as follows: one is that SACC uses the random grouping strategy while CCPA uses the proposed one, and the other is that SACC uses an auxiliary function to enhance its performance while CCPA does not. The algorithm MOS [28] is one of the best algorithms in CEC'2013 competition on large scale global optimization. The algorithm DECC-G [3] is used by many authors in their comparisons (e.g., [7, 28]), and the results has been given in [29] for CEC'2013 benchmark suite. Although a new method with differential grouping [7] has shown a good performance, it does not test CEC'2013 benchmark suite [19] and we cannot get their results on these benchmarks. Thus, we do not make the comparison with this new method. We directly use the results obtained by SACC, MOS and DECC-G in [27-28] and [3] for a comparison. These results are also given in Table 2.

It can be seen from Table 2 that, among 15 test functions, the mean values of seven functions obtained by CCPA are better than those obtained by other four algorithms, and the best values of eight functions obtained by CCPA are better than those obtained by other four algorithms, which indicates that CCPA are the most efficient algorithms among these compared algorithms for most functions.

For a pairwise comparison, both the mean values and the best values of nine functions obtained by CCPA are better than those obtained by MOS, which indicates that CCPA is more effective than MOS. The mean values of thirteen problems obtained by CCPA are better than those obtained by DECC-G and the best values of eleven functions obtained by CCPA are better than those obtained by DECC-G. The almost all results obtained by CCPA are better than those obtained by SACC and CCR, which indicates that the proposed problem adaptive grouping strategy PAVG is more effective than random grouping strategy.

For fully separable functions $f1, f2$ and $f3$, it can be seen from Table 2 that performance of CCPA is better than that of all the other four algorithms, which indicates that the proposed variable grouping strategy is more efficient.

For eight partially additively separable functions $f4 - f11$, the results obtained by CCPA are better than those obtained by CCR and SACC for almost all these functions. The performance of CCPA is better than that of MOS on five functions. The mean values of six partially additively separable problems $f4, f5, f7, f9, f10$ and $f11$ obtained by CCPA are better than those obtained by DECC-G, and the best solutions of five functions obtained by CCPA are better than those obtained by DECC-G. Thus, CCPA is more efficient on the most partially additively separable functions than other four compared algorithms. However, it can also be seen from Table 2 that for these eight partially additively separable functions $f4 - f11$, the best solutions obtained by all compared algorithms are far from the real global optimal solutions. This may be caused by following two reasons. One may be that the number of local minima grows exponentially as the number of decision variables increases; and second one may be that the maximum number of function evaluations assigned is not enough for these functions.

Finally, for overlapping functions $f12 - f14$ and non-separable function $f15$, the results in CCPA are poorer than those in MOS. It indicates that variable grouping strategy PAVG is ineffective for overlapping and non-separable functions, which is consistent with the designing of our algorithm.

In order to show the efficiency and effectiveness of the proposed variable grouping strategy intuitively, we use the semi-log line diagram to plot the convergence curve of the fitness value on Fig. 1 to Fig. 5, where the horizontal axis is the number of function evaluations and the vertical axis is logarithmic scale of the mean fitness value in 25 runs on a problem. Fig.1 to Fig. 5 show the convergence curves for the five representative functions: $f2, f3, f5, f7$ and $f10$, respectively, where the thick line represents the convergence curve of CCPA, and thin line represents the convergence curve of SACC, and dotted line represents the convergence curve of CCR.

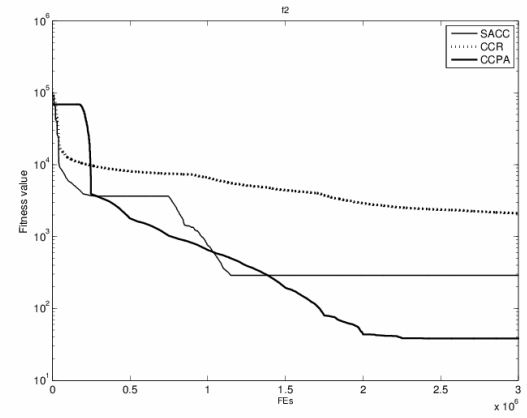

Fig. 1. The convergence curve of three algorithms on $f2$

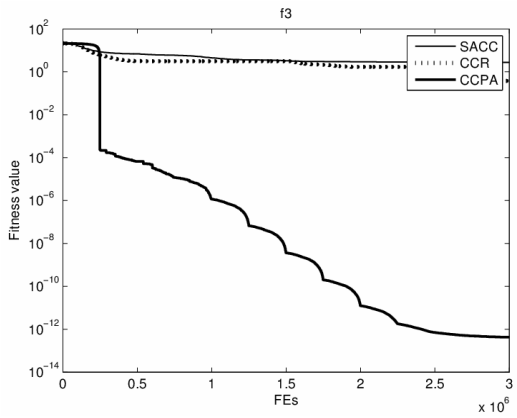

Fig. 2. The convergence curve of three algorithms on $f3$

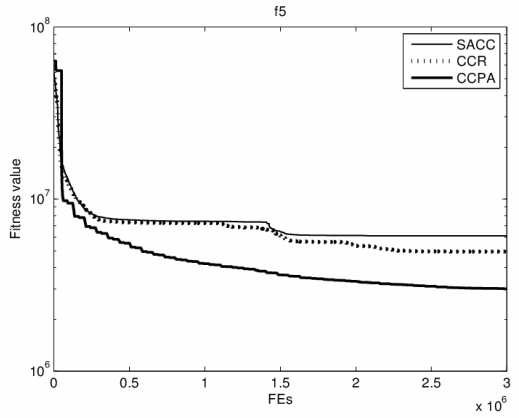

Fig. 3. The convergence curve of three algorithms on $f5$

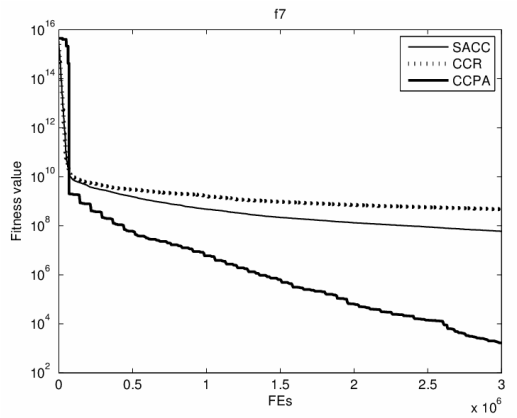

Fig. 4. The convergence curve of three algorithms on $f7$

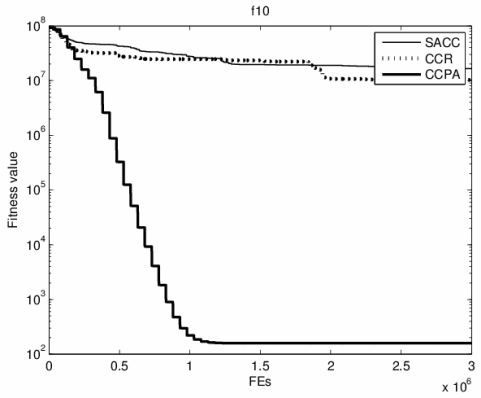

Fig. 5. The convergence curve of three algorithms on $f10$

From Fig. 1 to Fig. 5, it can be seen that CCPA converges much faster than the other two algorithms and more easily finds better solutions.

Overall, it can be seen from Tables 1, 2 and Fig. 1 to Fig. 5 that the problem adaptive variable grouping strategy PAVG plays an important part in CCPA and CCPA is more effective and efficient.

4 Conclusions

In this paper, a new evolutionary algorithm called cooperative co-evolution with problem adaptive variable grouping strategy (briefly, CCPA) has been proposed. First, the variable grouping strategy PAVG can more accurately group the interacting variables into the same subcomponent such that the interaction between different subcomponents is as minimum as possible. Second, a local search strategy has been given to enhance EAs. Finally, CC can search multiple regions simultaneously and thus has more possibility to find a better local optimal solution. In order to evaluate the actual performance of PAVG, we used PAVG in a cooperative co-evolutionary framework and conducted the experiments on 11 large-scale additively separable functions in CEC'2013 benchmark suite. The experiment results revealed that PAVG can accurately make the variable grouping and greatly enhance the performance of CC.

There are several relevant issues to be addressed further in the future. Firstly, more powerful variable grouping strategy for large scale global optimization needs to be further explored. Secondly, it is necessary to revise CCPA by integrating some other technique which can effectively handle the non-separable problems. Thirdly, our algorithm can only be applicable to the problems with their function equations being known. There are many problems that can be abstracted to the large-scale global optimization problems and the objective functions of these problems indeed exist, and CCPA is suitable for solving these kinds of large-scale problems.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. U1404622), and the Fund of Education Department of Shaanxi Provincial Government (No. 18JK0505), and the Cultivation Fund of Xi'an University of Science and Technology (No. 201644).

References

[1] M. Potter, K. Jong, A cooperative coevolutionary approach to function optimization, in: Proc. International Conference on Parallel Problem Solving from Nature, 1994.

[2] X. Li, X. Yao, Cooperatively coevolving particle swarms for large scale optimization, IEEE Transactions on Evolutionary Computation 16(2012) 210-224.

[3] Z. Yang, K. Tang, X. Yao, Large scale evolutionary optimization using cooperative coevolution, Information Sciences 178(2008) 2986-2999.

[4] Z. Yang, K. Tang, X. Yao, Multilevel cooperative coevolution for large scale optimization, in: Proc. IEEE Congress on Evolutionary Computation, 2008.

[5] W. Chen, T. Weise, Z. Yang, K. Tang, Large-scale global optimization using cooperative coevolution with variable interaction learning, PPSN 2(2010) 300-309.

[6] Y. Mei, X. Li, X. Yao, Cooperative co-evolution with route distance grouping for large-scale capacitated arc routing problems, IEEE Transactions on Evolutionary Computation 3(2014) 435-449.

[7] M. Omidvar, X. Li, Y. Mei, X. Yao, Cooperative co-evolution with differential grouping for large scale optimization, IEEE Transactions on Evolutionary Computation 3(2014) 378-393.

[8] M. Omidvar, X. Li, X. Yao, Cooperative co-evolution with delta grouping for large scale non-separable function optimization, in: Proc. IEEE Congress on Evolutionary Computation, 2010.

[9] B. Kazimipour, X. Li, A. Qin, Initialization methods for large scale global optimization, in: Proc. IEEE Congress on Evolutionary Computation, 2013.

[10] N. Dong, C. Wu, W. Ip, Z. Chen, C. Chan, K. Yung, An opposition-based chaotic GA/PSO hybrid algorithm and its application in circle detection, Computers & Mathematics with Applications 64(2012) 1886-1902.

[11] Y. Leung, Y. Wang, An orthogonal genetic algorithm with quantization for global numerical optimization, IEEE Transactions on Evolutionary Computation 5(1)(2001) 41-53.

[12] Z. Wu, N. Huang, A study of the characteristics of white noise using the empirical mode decomposition method, in: Proc. the Royal Society A: Mathematical, Physical & Engineering Sciences, 2004.

[13] R. Rach, J. Duan, Near-field and far-field approximations by the adomian and asymptotic decomposition methods, Applied Mathematics and Computation 217(2011) 5910-5922.

[14] S. Valdez, A. Hernández, S. Botello, A Boltzmann based estimation of distribution algorithm, Information Sciences 236(2013) 126-137.

[15] C. Ahn, J. An, J. Yoo, Estimation of particle swarm distribution algorithms: combining the benefits of PSO and EDAs, Information Sciences 192(2012) 109-119.

[16] G. Iacca, F. Neri, E. Mininno, Y. Ong, M. Lim, Ockham's Razor in memetic computing: three stage optimal memetic exploration, Information Sciences 188(2012) 17-43.

[17] F. Caraffini, F. Neri, G. Iacca, A. Mol, Parallel memetic structures, Information Sciences 227(2013) 60-82.

[18] 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, in: Proc. Nature Inspired Computation and Applications Laboratory, 2009.

[19] X. Li, K. Tang, M. Omidvar, Z. Yang, K. Qin, Benchmark functions for the CEC'2013 special session and competition on large scale global optimization, Technical Report, January 2013.

[20] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, KanGAL Report Number 2005005, May 2005.

[21] J.J. Liang, B.Y. Qu, Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization, Technical Report 201311, December 2013.

[22] J. Vesterstrom, R. Thomsen, A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical, in: Proc. the 2004 Congress on Evolutionary Computation, 2004.

[23] R. Gamperle, S. Muller, P. Koumoutsakos, A parameter study for differential evolution, in: Proc. WSEAS International Conference on Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, 2002.

[24] Z. Yang, K. Tang, X. Yao, Self-adaptive differential evolution with neighborhood search, in: Proc. IEEE World Congress on Computational Intelligence, 2008.

[25] J. Nocedal, S.J. Wright, Numerical Optimization, Springer-Verlag, New York, 1999.

[26] D. Molina, M. Lozano, F. Herrera, MA-SW-chains: Memetic algorithm based on local search chains for large scale continuous global optimization, in: Proc. WCCI 2010 IEEE World Congress on Computational Intelligence, 2010.

[27] F. Wei, Y.P. Wang, Y.L. Huo, Smoothing and auxiliary functions based cooperative coevolution for global optimization, in: Proc. IEEE Congress on Evolutionary Computation, 2013.

[28] A. LaTorre, S. Muelas, J.M. Pena, Large scale global optimization: experimental results with MOS-based hybrid algorithms, in: Proc. IEEE Congress on Evolutionary Computation, 2013.

[29] Preliminary result on DECC-G (a baselind model). <http://titan.csit.rmit.edu.au/~e46507/cec13-lsgo/competition/lsgo2013-decc-g.html>, 2013.