Journal of Network Intelligence ©2017 ISSN 2414-8105 (Online)
Taiwan Ubiquitous Information Volume **2**, Number **4**, November **2017**

A New Cooperative Co-evolution Algorithm Based on Variable Grouping and Local Search for Large Scale Global Optimization

Shiwei Guan

School of Computer Science and Technology
Xidian University
Xi'an, Shaanxi, 710071, China
BillJames_guan@163.com

Yuping Wang

School of Computer Science and Technology
Xidian University
Xi'an, Shaanxi, 710071, China
ywang@xidian.edu.cn

Haiyan Liu

School of Computer Science and Technology
Xidian University
Xi'an, Shaanxi, 710071, China
hyliu83@126.com

Received May 2017 Revised September 2017

Abstract. *Many evolutionary algorithms have been proposed for large scale global optimization, but there are still many difficulties when handling high-dimensional optimization problems because of the search space exponential growth with dimensionality. In this paper, we propose a hybrid algorithm with Cooperative Co-evolution (CC) framework to tackle this kind of problems. First, a formula based grouping strategy is adopted to classify the interacting variables into the same subcomponent, and all of the subcomponents are optimized with a Modified SaNSDE (M-SaNSDE) and Modified Solis Wet's algorithm (M-SW) iteratively in CC framework, and a new restart mechanism is proposed to help the population jump out of the local converge. We design a balance between exploration of M-SaNSDE to guide the population evolution direction and explorative of M-SW to optimize the individuals in the population. Experiment was carried out using a benchmark designed for large scale global optimization, showing that our algorithm obtains good results in most difficult functions compared with reference algorithms.*

Keywords: Large scale global optimization, Cooperative Co-evolution, Hybrid algorithm, Formula based grouping

1. Introduction.

Continuous optimization is an important research field because many real-world problems from different domains (engineering, economics, etc.) can be transformed as the optimization of a continuous functions. Optimization problems with a large number of variables are often referred as large scale optimization problems. Many Evolutionary Algorithms (EAs) have been proposed for this type of optimization problems, and they can get accurate solutions in complex problems without specific information about them, but the performance of these algorithms deteriorate as the dimensionality of the problems increases, commonly referred to as the curse of dimensionality [1].

In recent years, there are many new theoretical and computational contributions have been proposed for solving large scale global optimization (LSGO), e.g., cooperative co-evolut-ion (CC) [2–7], memetic algorithm [8–10], hybrid algorithms [11], decomposition methods [12–14], etc. For LSGO problems, one intuitive approach is to adopt a divide-and-conquer strategy. A typical kind of such approaches is cooperative co-evolution (CC) proposed by Potter and De Jong [2], it divides a big problem into several smaller sub-components, which referred to decision variables decomposition, each subcomponent is optimized using a separate EA in round robin fashion. Despite its success in solving many optimization problems, CC loses its efficiency when applied to problems which a proportion of the decision variables have interaction among themselves. When these interacting variables are decomposed into different subcomponents, the overall performance of the algorithm would be a major decline. This challenge calls new techniques which are capable of capturing the interacting variables and group them into one subcomponents.

In order to mitigate these problems, several effective grouping methods [7, 12–14] have been successively proposed. For example, Yang et al. proposed a random grouping method [7] which group the decision variables into different subcomponents randomly, the random grouping method increases the probability of two interacting variable allocated into the same subcomponent, but if the interacting variables are more than three, it can not group accurately. Mohammad and Li et al. proposed an automatic decomposition strategy called differential grouping [13] which group the decision variable via variables' difference, the experiments on CEC 2010 benchmark suite [15] have shown this grouping strategy is efficient and accurate, but with more complex CEC 2013 benchmark suite [16], this grouping strategy can not group the interacting decision variables into the same subcomponent accurately.

Another important consideration that greatly affects the overall optimization performance is the allocation of the available computational resources to various subcomponents. In a classic CC, all subcomponents receive an equal share of the available computational resources through the round-robin optimization of all subcomponents in each cycle. However, it has been shown that the round-robin strategy can waste a considerable amount of the available resources on the imbalanced functions [16], which refers to the unequal contribution of different subcomponents to the overall objective val. Omidvar et al [17] have shown that in the presence of strong imbalance in the contribution of subcomponents to the overall objective value, only a few subcomponents have dominant effect on the overall improvement of the objective val, while making the contribution of the other subcomponents negligible. In such situations, most of the efforts in the optimization of the subcomponents with a lower contribution is wasted. To alleviate the imbalance issue, Contribution-Based Co-operative Co-evolution(CBCC) [17] has been proposed in which the subcomponents with a higher contribution are given a higher share of the available resources. While the subcomponents' contribution maybe unknown to the optimizer, CBCC adopts rules to approximate these values and dynamically select the most contributing subcomponents for the next round of optimization.

In this work we present a new hybrid evolutionary algorithm based on CC framework for large scale global optimization. The main idea is to group the decision variables via an accurate decomposition strategy—Formula Based Gro-uping strategy (FBG) [18, 19], then evolve the subcomponents though modified SaNSDE [20] (M-SaNSDE) and modified Solis and Wet's local search method [21] (M-SW) in a round robin fashion, combining the exploratory of the M-SaNSDE with the exploitative factor of the M-SW method. Our proposal is characterized by several features:

First, FBG strategy is adopted to group the decision variables into several subcomponents, the FBG strategy costs lower computing resource and gets accurate grouping result.

Second, a modified DE is proposed (M-SaNSDE) in order to avoid function evaluations (FEs) cost without function value changed and guide the overall search direction.

Third, a modified LS method (M-SW) is proposed to optimize the selected individuals' largest contribuation subcomponent exploitatively without care of premature converge.

Fourth, a balance between M-SaNSDE in guiding the search direction and M-SW method in optimizing the individuals exploitatively is constructed in every evolving cycle.

Finally, a new restart mechanism is designed to help the population escape from local converge.

The remainder of the paper is organized as follows: Section 2 describes the preliminaries and background information; The proposed algorithm will be described in detail in Section 3; Section 4 demonstrates and analyzes the experimental results and finally Section 5 summarizes this paper.

2. **Problem Statement and Preliminaries.**

We consider the following global optimization problem:

$$\begin{cases} \min \ f(\boldsymbol{x}) \\ s.t \ \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u} \end{cases}$$

where $\boldsymbol{x} = (x_1, x_2, \cdots, x_N)$ is a vector in R^N, $f(\boldsymbol{x})$ is the objective function, $\boldsymbol{l} = (l_1, l_2, \cdots, l_N)$, and $\boldsymbol{u} = (u_1, u_2, \cdots, u_N)$ with $x_i \in [l_i, u_i]$ for $i = 1 \sim N$.

Definition 1: A function is separable if:

$$arg \min_{(x_1,x_2,x_3,\cdots,x_n)} f(x_1, x_2, x_3, \cdots, x_n) = (\underset{x_1}{\text{argmin}} f(x_1, \cdots), \quad \cdots, \underset{x_n}{\text{argmin}} f(\cdots, x_n))$$

And its global optimum can be reached by successive line search along axes. If a certain function is not separable, there must be interaction between at least two variables in the decision vector. So if it is possible to find the global optimum of a function by optimizing one dimension at a time regardless of the values taken by other dimensions, then the function is said to be separable, otherwise, it is said to be non-separable.

2.1. **Cooperative Co-evolution.**

Cooperative co-evolution (CC) is a framework to decompose a large-scale problem into a set of smaller sub-problems, then evolved each of the sub-problems using a separate EA. The critical of CC can be summarized as follows:

1. Problem decomposition: decompose the high-dimensional problem into several smaller subcomponents in which each subcomponent can be handled by a certain EA.
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.

In the original implement of cooperative co-evolution, Potter and De Jong [4] incorporate CC with a genetic algorithm and decomposed an n-dimensional problem into n 1-D problem, and the problem only have a maximum of 30 dimensions. Liu et al. [22] made the first attempt to solve the large-scale optimization problem using CC framework, they adopted fast evolutionary programming with CC on benchmark problems up to 1000 dimensions, but they handle the large scale problem without care about variable interaction. Cooperative Particle Swarm Optimization (CPSO) [23] is the first attempt to incorporate PSO with CC framework, but there are many parameters to be set in the particle swarm optimization.

2.2. **Decomposition strategies.** Cooperative Co-evolution is a good framework for large scale global optimization, but there are also many problems when using it. One drawback of CC is its sensitive to the choice of decomposition strategy, the performance of CC changing dramatically when incorporated with different grouping strategy.

The goal of the grouping strategy is to divide the decision variables with variable interaction into the same subcomponent, and the decision variables without interaction are not grouped into the same subcomponent. With the large scale global optimization get more and more attention, there are many new grouping strategies have been proposed, such as route distance grouping [5], MLCC [6], random grouping [7], variable interaction learning grouping (CCVIL [12]), differential grouping [13], delta grouping [14], and so on. However, most of these grouping strategies divide a n-dimension problem into k s-dimension sub-problems with no care about variable interaction and can not group the decision variables accurately. For example, random grouping decomposes a problem into k s-dimension sub-problem, and it randomly allocates the decision variables to subcomponents in every co-evolutionary cycle. It was shown mathematically that with the random grouping the possibility of placing two interacting variables in the same subcomponent for several cycle is reasonably high, but the possibility drops dramatically when the interacting variables is more than three. In MLCC, instead of using a fixed number for s, a set of possible s values is provided to the algorithm. During the course of evolution, the performance of each subcomponent size is measured and the value with better performance are given a higher possibility of being selected in the next co-evolution cycle. This technique partially solves the problem with a set of s value, but it is hard for user to choose s value in practice, and it groups the decision variables with no care about the variable interaction.

To overcome the shortcomings of ignoring the decision variables interaction and fixed subcomponent size, the formula based grouping (FBG) strategy [18, 19] is adopted. FBG can identify the interacting variables accurately and decompose them into the same subcomponent. The FBG framework works as follows:

Firstly, build a set N-sep whose elements result in variables interaction;

Secondly, search the variables in the operations in the N-sep in the expression of the objective function;

Finally, group the interaction variables into the same subcomponent.

Note that an expression of a general objective function consists of finite number of four arithmetic operations '+', '−', '×', '÷' and composite operations elementary functions. And the N-sep is constructed as follows:

(1) Variable separability in four arithmetic operations. Variables connected by '+' and '−' are separable while non-separable by '×', '÷';

(2) Composite of basic elementary function

$$g(h(x)), x = (x_1, x_2, x_3, \cdots, x_n)$$

when $g(x)$ is monotonous and variables in $h(x)$ is separable, variable in $g(h(x))$ still separable, otherwise non-separable.

(3) Composite functions combined with the four arithmetic operators. a) '+' and '−' won't change the separability of the composite functions. b) For '×' and '÷', the variables in the composite functions are all non-separable except that both composite functions are exponential functions.

2.3. **Self-adaptive Differential Evolution with Neighborhood Search.** Individual in differential evolution (DE) are represent by D-dimensional vector $x_i, \forall i \in \{1, \cdots, NP\}$, where D is the number of decision variables and NP is the population size. According to [24], the classical DE can be summarized as follows:

Algorithm 1 FBG algorithm

```
1. Construct N-sep according to the above rules;
2. Match the function expression with N-sep elements, put the interacting variables
   into the same subcomponent;
3. Repeat step 2 until all variables linked by elements of N-sep are checked;
4. The variables which are not grouped in step 2 and 3 are grouped into a separate
   subcomponent;
```

1)Mutation:

$$v_i = x_{i_1} + F \cdot (x_{i_2} - x_{i_3})$$

Where $i_1, i_2, i_3 \in [1, NP]$ are random and mutually different integer. Scale factor $F > 0$ is real constant and is often set to 0.5.

2)Crossover:

$$u_i(j) = \begin{cases} u_i \ , & \text{if } u_j(0,1) \le CR \ or \ j == j_{rand} \\ x_i \ , & \text{otherwise} \end{cases}$$

With $u_j(0,1)$ stands for the uniform random number between 0 and 1,and j_{rand} is a randomly chosen index to ensure that the trial vector u_i does not duplicate x_i ,$CR \in (0,1)$ is the crossover rate, which is often set to 0.9.

3)Selection:

$$x_i^{'} = \begin{cases} u_i \ , & \text{if } f(u_i) \le f(x_i) \\ x_i \ , & \text{otherwise} \end{cases}$$

The selection operation will choose the individual with better function value as the offspring from current x_i and u_i.

There are several schemas of DE based on different mutation strategies:

$$v_i = x_{i_1} + F \cdot (x_{i_2} - x_{i_3}) \tag{1}$$

$$v_i = x_{best} + F \cdot (x_{i_1} - x_{i_2}) \tag{2}$$

$$v_i = x_{i_1} + F \cdot (x_{best} - x_i) + F \cdot (x_{i_1} - x_{i_2}) \tag{3}$$

$$v_i = x_{best} + F \cdot (x_{i_1} - x_{i_2}) + F \cdot (x_{i_3} - x_{i_4}) \tag{4}$$

$$v_i = x_{i_1} + F \cdot (x_{i_2} - x_{i_3}) + F \cdot (x_{i_4} - x_{i_5}) \tag{5}$$

Where x_{best} represents the individual has the best function value. x_{i_1} ,x_{i_2} ,x_{i_3} ,x_{i_4} is randomly selected from the population and different from each other. Schemas (1) and (3), with notations as DE/rand/1 and DE/current to best/2, are the most used in practice due to their good performance.

Self-adaptive Differential Evolution (SaDE) [25] is a variant of DE which pays special attention to self-adaption between different mutation strategies as well as the self-adaption on crossover rate CR, while NSDE [26] intends to mix search biases of different neighborhood search operators though parameter F, but with no self-adaption for the parameters.

SaNSDE is an integration of Self-adaptive Differential Evolution (SaDE) and NSDE works as follows:

1)Mutation strategies self-adaption:

$$D = \begin{cases} \text{Eq.(1)} \ , & \text{if } u_j(0,1) < p \\ \text{Eq.(3)} \ , & \text{otherwise} \end{cases}$$

Where p is set to a constant initially, After evaluation of all offspring, the number of offspring generated by Eq.(1) and Eq.(3) successfully entering the next generation are recorded as ns_1 and ns_2 , while discarded are recorded as nf_1 and nf_2 . And those two pairs of number are accumulated within a specified number of generation called "learning period", and probability is updated as:

$$p = \frac{ns_1 \cdot (ns_2 + nf_2)}{ns_2 \cdot (ns_1 + nf_1) + ns_1 \cdot (ns_2 + nf_2)}$$

2)Crossover with Scale factor F self-adaption:

$$F_i = \begin{cases} N_i(0.5, 0.3) \ , & \text{if } u_i(0,1) \le fp \\ \delta_i \ , & \text{otherwise} \end{cases}$$

Where fp will be self-adapted as p done in SaDE.
3)Selection is the same with SaDE:

$$x_i{}' = \begin{cases} u_i \ , & \text{if } f(u_i) \le f(x_i) \\ x_i \ , & \text{otherwise} \end{cases}$$

3. **Proposed Algorithm.**

3.1. **Modified SaNSDE.**

SaNSDE is a good evolution algorithm for large scale problem, which can be applied to many different problems and get good results because of its parameter self-adaptive. But there are also some aspects can be improved. For example, we always use a fixed number of function evaluations to evolve the problem in a cycle, when the problem gets trapped into local converge, the function evaluations will be wasted in the following evolving process, we improve the algorithm from this aspect so that the function evaluation can be fully used in the evolving process. We set a threshold *epsilon* which can be adjusted dynamically according to the *bestval* obtained by the individual optimized with the best function value, and the *epsilon* is one in a thousand of *bestval*. If the difference between *bestval* and *prevbest* is continuous less than *epsilon* for a fixed times, we assume that the the computational resources will be wasted in the next optimization, and the optimization trapped into the local convergence, this Sub-dimension group optimization will broken and continue to next sub-dimension group optimization, thus the saved computational resources will be used to make more effects in the optimization.This analyzing mechanism will save the function evaluation from being wasted in the optimization with no function value change. The algorithm framework is shown in Algorithm 2, and the meaning of the notations used in the Algorithm 2 is as follows:
itermax: the total number of function evaluations for one optimization cycle;
bestval: is the current global optimum of the population;
epsilon: the threshold dynamic set according to *bestval*;
breakdata: the threshold to break the optimization process;

3.2. **Modified Solis Wet's algorithm.**

Traditional Solis Wet's algorithm is a randomised hill-climber with an adaptive step size, it explores all decision variables at the same time and modifies the current solutions by a difference vector randomly generated by Gaussian distribution and a fixed bias vector, it has good effect when handling non-separable functions (usually the dimension of the problem is low). But when dealing with high dimensional problem, for example, the dimension of the problem is 1000, Solis Wet's algorithm performance is poor and will cost many function evaluations without obvious effect.

In order to improve the performance of Solis and Wet's Algorithm for high dimensional problem, a new *biasvector* is added, the traditional Solis and Wet's algorithm uses a fixed bias vector to increase the possibility of finding a better result, but the high dimensional problem's search space is so big that a fixed difference vector is not suitable for different

Algorithm 2 Modified SaNSDE algorithm

```
while $numEval < itermax$ do
    $prevbest = bestval$;
    Mutation;
    Crossover;
    Selection;
    Set $epsilon = p * bestval$;
    if $prevbest - bestval < epsilon$ then
        $count = count + 1, cocount = cocount + 1$;
    else
        $count = 0, cocount = cocount + 1$;
    end if
    if $count > breakdata$ then
        break;
    end if
end while
if $count > breakdata$ then
    $usedFEs = cocount$;
else
    $usedFEs = itermax$;
end if
```

function landscape, so a dramatic *biasvector* is incorporated into the algorithm to diversity the descent direction, which can be more effective when the search space is big. The bias is generated in this way: sort the population based on the function value, and an individual randomly selected from %p best individuals [27] in the population called $p1$, another individual randomly selected from the rest population called $p2$, *biasvector* is the difference vector of $p1$ and $p2$, and $p1$, $p2$ will selected every cycle, and more search direction will be exploited to find the better objective val. Because of the high dimension, we divide the decision variables with FBG into several subcomponents, after optimization with M-SaNSDE, the first three subcomponents with the biggest improvement would be explored by modified Solis and Wet's algorithm (M-SW) every cycle successively, the function evaluations can be fully used, and M-SW explores part of the decision variables, which can avoid solving the high dimensionality problem directly as well as using the exploration property of the algorithm, and also becomes more useful to handle dimension reduction problem. The algorithm framework is shown in Algorithm 3, and the meaning of the notations used in the Algorithm 3 is as follows:
biasvector: the bias generated by two individuals selected from population;
subpop_index: the sub-dimensions selected to optimize;

3.3. **Restart Mechanism.** An interesting element in our proposal is restart mechanism, when the problem is very complex and there are many local optimums, the optimization procedure usually get trapped into local converge, which will cost many function evaluations with no effect. How to jump out of the local optimum is crucial, we design a new restart mechanism to decide whether current solution is struck in a local optimum, we compare the best function value *bestval* obtained from the current optimization and previous *prevbest*, if the difference of these two value is less than a threshold obtained from *bestval* in continuous optimization stage, we assume that current population get trapped into local convergence, we will choose two individuals and spread points around them with uniform distribution to generate a new population. The restart mechanism

Algorithm 3 Modified Solis and Wet's algorithm

```
Initialize $numSuccess$,$numFailed$,$numEval$ to 0;
while $numEval < maxEval$ do
    $diff(subpop\_index)$ = randomGaussian(0,rho);
    generate $p1$,$p2$ from population;
    $biasvector = rand() * (p1 - p2)$;
    $sol' = sol + biasvector + diff$;
    if $sol'$ better than $sol$ then
        $numSuccess = numSuccess + 1, numFailed = 0$;
    else
        $sol' = sol - biasvector - diff$;
        if $sol'$ better than $sol$ then
            $numSuccess = numSuccess + 1, numFailed = 0$;
        else
            $numFailed = numFailed + 1, numSuccess = 0$;
        end if
    end if
    if $numSuccess > 5$ then
        rho =2*rho,$numSuccess = 0$;
    else
        rho =rho/2,$numFailed = 0$;
    end if
    $numEval = numEval + 1$;
end while
```

helps the algorithm to jump out of the local optimum. The restart mechanism is shown in Algorithm 4, and the meaning of the notations used in the Algorithm 4 is as follows:
MAXVAL: the total function evaluation;
*breakdata*1: the threshold for testing whether the population get trapped into local convergence;

3.4. **Balance between M-SaNSDE and M-SW.** We use M-SaNSDE to optimize all of the dimension subcomponents and record the improvement value of each subcomponents, after one cycle of M-SaNSDE optimization, we can guide the overall search direction, with the record of improvement, the largest three contribution subcomponents is found and more computational resource can be allocated for to optimize these subcomponents with M-SW to exploit the search landscape. Because of the problem is imbalance, the larger contribution, the more fine optimization with M-SW can be put. The exploratory of the M-SaNSDE can be used for the exploitative factor of the M-SW.

3.5. **CC-FBG-DELS.** In this paper we proposed a hybrid evolution algorithm with CC framework (CC-FBG-DELS).

Firstly, FBG grouping strategy is used to group a complex large-scale problem's decision variables into several accurate subcomponents with no function evaluation. Secondly, (M-SaNSDE) is used to guide the population evolutionary direction, (M-SW) is used to exploit the individuals with two individuals' top 3 largest improvement subcomponents after M-SaNSDE algorithm. Finally, if the population get trapped into local converge, the restart mechanism would be used to help the population jump out of the local optimum. In CC-FBG-DELS, we use M-SaNSDE to optimize the whole population and guide the evolution direction, and M-SW is used with only two individuals' top 3 sub-dimension group with

Algorithm 4 Restart Mechanism

```
while $usedFEs <$ MAXVAL do
    $tempbestval = bestval$;
    optimization procedure;
    if $prevbest - bestval < epsilon$ then
        $count = count + 1$;
    else
        $count = 0$;
    end if
    if $count > breakdata1$ then
        break;
    end if
    if $count > breakdata$ then
        sort population based on function value;
        select two individuals from population, one from the top half of the population according to function value , and the other from last half of the population according to function value
        use uniform distribution to spread point around them;
        put the best individual into the population;
    end if
end while
```

largest improvement, so that we can make precise search against the important part of the search landscape, the exploratory of the M-SaNSDE and explorative of M-SW can be fully used in the optimization procedure. The CC-FBG-DELS algorithm framework is shown in Algorithm 5:

Algorithm 5 CC-FBG-DELS

```
1: Use FBG-grouping strategy to group the dimensions into different group;
2: Initialize the population;
3: Evaluate the population;
4: while $usedFEs < MAXVAL$ do
5:     Use M-SaNSDE to optimize each sub-dimension group and record their improvement values compared with the latest optimization;
6:     Select two individuals, one is the individual with best function value, another is the randomly selected from the rest of the population;
7:     Optimize the two individuals' top three sub-dimension groups with the largest improvement values recorded in the Cooperative Co-evolution process by M-SW with dynamic threshold, and judge if the sub-dimension group need one more cycle to optimize. if so, optimize the same sub-dimension group another cycle;
8:     Use Restart Mechanism to decide whether the population need update;
9: end while
```

4. Numerical Experiment.

4.1. Experimental Setup.

4.1. Experimental Setup. We evaluate the performance of the proposed CC-FBG-DELS algorithm on the test suite provided by CEC's 2013 special session on Large Scale Global Optimization [16]. This benchmark is composed by 15 continuous optimization functions for dimension 1000 and with different degrees of separability, from completely

TABLE 1. Parameter values used in CC-FBG-DELS

Parameter	Description	Value
popsize	population size	60
FEs	total function evaluation	3.0e+6
FE-M-SW	evaluation for each M-SW run	150
FE-M-SaNSDE	evaluation for each M-SaNSDE run	100

separable function to fully non-separable functions:
Fully separable function: $f_1 - f_3$
Partially separable functions: with a separable sub-component $f_4 - f_7$, with no separable subcomponent $f_8 - f_{11}$
Overlapping functions: $f_{12} - f_{14}$
Non-separable functions: f_{15}
The parameters used in our proposal are indicated in Table 1.

4.2. **Experimental Result on CEC Benchmark 2013.** Table 2 shown the result obtained from CC-FBG-DELS compared with other results obtained from the referenced algorithms DECC-CG [7], MOS [28]. DECC-CG is the initial algorithm tested on Benchmark 2013 [16], it uses random grouping with CC framework. MOS is the current best algorithm tested on Benchmark 2013 and get the best results on the whole, it uses many different algorithms together to evolve the population without grouping strategy.

From table 2 we can observe that our algorithm can optimize the problem f_1 nearly to zero, but has no good results compared with MOS and DECC-CG in fully separable problems f_2-f_3. For partially separable problems $f_4 - f_7$, there is a separable subcomponent, CC-FBG-DELS outperforms MOS and DECC-CG on 3 problems, especially for problem f_7, the optimization result nearly to zero, where MOS's result is still on 1e+04 scale. For problems $f_8 - f_{11}$, there is no separable subcomponent, CC-FBG-DELS outperforms MOS and DECC-CG on the other three problems, and as for f_{10}, the above three algorithm get the result almost the same. For overlapping problems $f_{12} - f_{14}$, CC-FBG-DELS has no particularly good result than MOS, but the results are still better than DECC-CG except on f_{12}. For non-separable problem f_{15}, CC-FBG-DELS has better result than MOS and DECC-CG. Overall, CC-FBG-DELS is better than MOS and DECC-CG when handling partially separable and non-separable problems, and the ability to deal with fully separable and overlapping problems needs to be improved.

5. **Conclusions.** In this work we proposed a new algorithm for large scale global optimization by using FBG grouping strategy to group the decision variables, and evolve the population with M-SaNSDE and M-SW iteratively. We construct a balance between the exploration of the M-SaNSDE to guide the population evolution direction and the explorative of M-SW to optimize the individuals in the population. There are several relevant issues to be addressed further in the future.

Firstly, our algorithm's performance on overlapping problems performs unsatisfactorily, which illustrates that we should do more research to make our algorithm more efficient when handing different type of large scale problems. Secondly, the Benchmark we used in experiment is not real problem, we should make our algorithm tested on the real problem to show its practicability.

Acknowledgment. This work was supported by National Natural Science Foundation of China (No. 61472297 and U1404622).

Table 2. Comparison of the Results with Reference algorithms

		f1	f2	f3	f4	f5
CC-FBG-DELS	Best	1.34E-24	1.02E+03	2.10E+01	**2.02E+07**	**2.87E+06**
	Median	2.03E-22	1.19E+03	2.11E+01	**3.74E+07**	**4.11E+06**
	Worst	1.18E-21	1.35E+03	2.11E+01	**1.44E+07**	**6.13E+06**
	Mean	2.71E-22	1.19E+03	2.11E+01	**4.67E+07**	**4.16E+06**
	Std	2.68E-22	9.07E+01	1.67E-02	**2.68E+07**	**8.55E+05**
DECC-DG	Best	1.75E-13	9.90E+02	2.63E-10	7.58E+09	7.28E+14
	Median	2.00E-13	1.03E+03	2.85E-10	2.12E+10	7.28E+14
	Worst	2.45E-13	1.07E+03	3.16E-10	6.99E+10	7.28E+14
	Mean	2.03E-13	1.03E+03	2.87E-10	2.60E+10	7.28E+14
	Std	1.78E-14	**2.26E+01**	1.38E-11	1.47E+10	1.51E+05
MOS	Best	**0.00E+00**	**7.40E+02**	**8.20E-13**	1.10E+08	5.25E+06
	Median	**0.00E+00**	**8.36E+02**	**9.10E-13**	1.56E+08	6.79E+06
	Worst	**0.00E+00**	**9.28E+02**	**1.00E-12**	5.22E+08	8.56E+06
	Mean	**0.00E+00**	**8.32E+02**	**9.17E-13**	1.74E+08	6.94E+06
	Std	**0.00E+00**	4.57E+01	**5.23E-14**	8.03E+07	9.03E+05
		f6	f7	f8	f9	f10
CC-FBG-DELS	Best	1.04E+06	**6.51E-01**	**1.22E+11**	**2.34E+08**	9.07E+07
	Median	1.06E+06	**1.62E+00**	**1.78E+12**	**3.39E+08**	9.42E+07
	Worst	1.06E+06	**5.66E+00**	**1.03E+13**	**4.38E+08**	9.47E+07
	Mean	1.06E+06	**2.02E+00**	**2.84E+12**	**3.32E+08**	9.39E+07
	Std	5.09E+03	**1.31E+00**	**2.78E+12**	6.57E+07	1.02E+06
DECC-DG	Best	**6.96E-08**	1.96E+08	1.43E+14	2.20E+08	9.29E+04
	Median	**6.08E+04**	4.27E+08	3.88E+14	4.17E+08	1.19E+07
	Worst	**1.10E+05**	1.78E+09	7.75E+14	6.55E+08	1.73E+07
	Mean	**4.85E+04**	6.07E+08	4.26E+14	4.27E+08	1.10E+07
	Std	**3.98E+04**	4.09E+08	1.53E+14	9.89E+07	4.00E+06
MOS	Best	1.95E+01	3.49E+03	3.26E+12	2.63E+08	**5.92E+02**
	Median	1.39E+05	1.62E+04	8.08E+12	3.87E+08	**1.18E+06**
	Worst	2.31E+05	3.73E+04	1.32E+13	5.42E+08	**1.23E+06**
	Mean	1.48E+05	1.62E+04	8.00E+12	3.83E+08	**9.02E+05**
	Std	6.56E+04	9.29E+03	3.14E+12	**6.42E+07**	**5.17E+05**
		f11	f12	f13	f14	f15
CC-FBG-DELS	Best	**6.17E+05**	2.58E+03	2.67E+08	1.03E+08	**1.03E+06**
	Median	**1.40E+06**	2.94E+03	7.94E+08	3.14E+08	**1.28E+06**
	Worst	**2.99E+06**	3.78E+03	1.74E+09	1.14E+10	**1.79E+06**
	Mean	**1.49E+06**	3.00E+03	8.46E+08	1.31E+09	**1.30E+06**
	Std	**6.97E+05**	2.93E+02	4.51E+08	2.65E+09	**1.70E+06**
DECC-DG	Best	4.68E+10	9.80E+02	2.09E+10	1.91E+11	4.63E+07
	Median	1.60E+11	1.03E+03	3.36E+10	6.27E+11	6.01E+07
	Worst	7.16E+11	1.20E+03	4.64E+10	1.04E+12	7.15E+07
	Mean	2.46E+11	1.04E+03	3.42E+10	6.08E+11	6.05E+07
	Std	2.03E+11	**5.76E+01**	6.41E+09	2.06E+11	6.45E+06
MOS	Best	2.06E+07	**2.22E-01**	**1.52E+06**	**1.54E+07**	2.03E+06
	Median	4.48E+07	**2.46E+02**	**3.30E+06**	**2.42E+07**	2.38E+06
	Worst	9.50E+07	**1.17E+03**	**6.16E+06**	**4.46E+07**	2.88E+06
	Mean	5.22E+07	**2.47E+02**	**3.40E+06**	**2.56E+07**	2.35E+06
	Std	2.10E+07	2.59E+02	**1.08E+06**	**8.11E+06**	1.98E+05

REFERENCES

[1] M. N. Omidvar, X. Li, Z. Yang, and X. Yao, Cooperative co-evolution for large scale optimization through more frequent random grouping, pp. 1–8, July 2010.

[2] M. A. Potter and K. A. D. Jong, Cooperative coevolution: An architecture for evolving coadapted subcomponents, *Evolutionary Computation*, vol. 8, pp. 1–29, 2000.

[3] M. Potter and K. Jong, Cooperatively coevolving particle swarms for large scale optimization, 1994.

[4] M.Potter and K.Jong, A cooperative coevolutionary approach to function optimization, Proceedings of International Conference on Parallel Problem Solving from Nature, 1994.

[5] Y. Mei, X. Li, and X. Yao, Cooperative co-evolution with route distance grouping for large-scale capacitated arc routing problems, 2013.

[6] Z. Yang, K. Tang, and X. Yao, Multilevel cooperative coevolution for large scale optimization, Proceedings of IEEE Congress on Evolutionary Computation, 2008.

[7] Z.Yang, K.Tang, and X.Yao, Large scale evolutionary optimization using cooperative coevolution, 2008.

[8] D. Molina and F. Herrera, Iterative hybridization of de with local search for the cec'2015 special session on large scale global optimization, pp. 1974–1978, May 2015.

[9] M. H. F. Molina, D.and Lozano, Ma-sw-chains: Memetic algorithm based on local search chains for large scale continuous global optimization, pp. 1–8, July 2010.

[10] D. Molina, M. Lozano, and F. Herrera, Ma-sw-chains: Memetic algorithm based on local search chains for large scale continuous global optimization, 2010.

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

[12] W. Chen, T. Weise, Z. Yang, and K. Tang, Large-scale global optimization using cooperative coevolution with variable interaction learning, PPSN, 2010.

[13] M. Omidvar, X. Li, Y. Mei, and X. Yao, Cooperative co-evolution with differential grouping for large scale optimization, 2013.

[14] M. Omidvar, X. Li, and X. Yao, Cooperative co-evolution with delta grouping for large scale non-separable function optimization, Proceedings of IEEE Congress on Evolutionary Computation, 2010.

[15] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, Benchmark functions for the cec'2010 special session and competition on large-scale global optimization, tech. rep., Nature Inspired Computation and Applications Laboratory , USTC, China, 2009.

[16] X. Li, K. Tang, M. Omidvar, Z. Yang, and K. Qin, Benchmark functions for the cec'2013 special session and competition on large scale global optimization, tech. rep., RMIT University, 2013.

[17] X. L. M. N. Omidvar and X. Yao, Smart use of computational resources based on contribution for coop-erative co-evolutionary algorithms, 2011.

[18] F. Wei, Y. Wang, and T. Zong, A novel cooperative coevolution for large scale global optimization, in *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, pp. 738–741, Oct 2014.

[19] F.Wei, Y.Wang, and T.Zong, Variable grouping based differential evolution using an auxiliary function for large scale global optimization, in *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pp. 1293–1298, July 2014.

[20] Z. Yang, K. Tang, and X. Yao, Self-adaptive differential evolution with neighborhood search, Proceedings of IEEE World Congress on Computational Intelligence, 2008.

[21] R. J.-B. W. Francisco J. Solis, Minimization by random search techniques, *Mathematics of Operations Research*, vol. 6, no. 1, pp. 19–30, 1981.

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

[23] F. van den Bergh and A. P. Engelbrecht, A cooperative approach to particle swarm optimization, *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 225–239, June 2004.

[24] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*. Springer-Verlag New York, Inc., 2005.

[25] A. K. Qin and P. N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in *2005 IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1785–1791 Vol. 2, Sept 2005.

[26] Z. Yang and X. Yao, Making a difference to differential evolution, Springer Berlin Heidelberg, 2008.

[27] J. Zhang and A. C. Sanderson, Jade: Adaptive differential evolution with optional external archive, *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 945–958, Oct 2009.

[28] S. Muelas and J. Pe, Large scale global optimization: Experimental results with mos-based hybrid algorithms, Proceedings of IEEE Congress on Evolutionary Computation, 2013.