Cooperative Co-evolution with Graph-based Differential Grouping for Large Scale Global Optimization

Yingbiao Ling and Haijian Li
School of Data and Computer Science,
Sun Yat-sen University,
Guangzhou, P.R. China

Bin Cao
School of Computer Science and Engineering,
Hebei University of Technology,
Hebei Province Key Laboratory of Big Data Calculation,
Tianjin, P.R. China
caobin@scse.hebut.edu.cn

***Abstract*—Cooperative co-evolution framework has been developed for solving large scale global optimization problems. This approach applies the divide-and-conquer strategy that decomposes the problem into subcomponents which can be optimized separately. Nevertheless, the decomposition strategy has important influence on solution quality. In theory, the interdependency between subcomponents should be kept minimum as the subcomponents coadaptation is needed to solve large scale optimization problems. Some state of the art decomposition strategies like differential grouping gain high grouping accuracy on a suite of benchmark functions. In this paper, we use graph theory to model the decomposition problem and propose graph-based differential grouping decomposition strategy to improve the decomposition accuracy of differential grouping. Empirical studies show that our decomposition method get perfect performance on all benchmark functions. Significantly, the solution quality on large scale problems are better than several outstanding decomposition strategies when the graph-based differential grouping is embedded with cooperative co-evolution framework.**

I. Introduction

In the past decades, optimization problems are becoming increasingly complex: non-differentiable, discontinuous are emerging constantly. Evolutionary algorithms (EAs), which is a set of algorithms based on biological evolution, show excellent performance on these problem. However, the performance of many EAs is unsatisfactory when the dimensionality of the problem is relatively large[1]. Cooperative co-evolution (CC) [2] is an efficient framework proposed by Potter and De Jong in solving large scale optimization problem. CC can be worked with an EA to tackle optimization problems, this kind of algorithm is called CCEA. CCEA solves the whole problem by optimizing subproblems separately by an EA once the decomposition is accomplished. It was shown that this divide-and-conquer strategy was very useful in solving many separable problems when combined with genetic algorithm [3]. However, it was struggling when solving non-separable problems. The main reason for these results was CC did not consider variable interaction during decomposition procedure.

Recently, plenty of research results have shown that the performance of CCEA is strongly associated with decomposition strategy [4]. Ideally, interacting variables should be allocated into the same subcomponent and dependence between different subcomponents is supposed to keep minimum. As there is no variable interaction in separable functions, separable function can be solved by optimizing one dimension while setting other dimensions to constant values. However, variable interaction in nonseparable functions have great influence on final solution quality. Therefore, further work would be needed to identify variable interaction in order to enhance the performance of CCEA on nonseparable problems[1].

In recent years, researchers have proposed several decomposition strategies and authenticated their effectiveness [5–8]. The differential grouping (DG) proposed by Omidvar et al. show excellent performance on most benchmark functions [8]. Nevertheless, DG does not perform well on the all the instances of Rosenbrock function [9]. It indicates that the grouping accuracy of DG can be further improved.

In this paper, we investigate the interaction between variables and the performance of several state-of-the-art decomposition methods within CC framework. We focus on analyzing the weakness of DG and find that the main rules used in DG are effective in identifying pairwise interaction, but DG may lose its effectiveness when identifying the interaction among multiple variables, which results in undesirable performance on the instance of Rosenbrock function. Y. Sun et al analyze the performance of DG and proposed extend differential grouping (XDG) [10], but XDG did not model the decomposition problem in the formal way.

We propose an graph-based differential grouping (gDG) methods to further improve the performance of DG. Firstly, gDG identifies all separable variables and allocates them in the same group. Secondly, considering each variable as a vertex and interaction between two variables as an edge, gDG constructs a graph and identifies connected components in such a graph. After allocating the variables in the same connected components in the same subcomponent, we find that gDG can improve the grouping accuracy greatly. Benchmark functions from CEC'2010[1] are used to test the efficiency of gDG. The experimental results shows that gDG obtains better decomposition results than original DG, especially on the in-

978-1-5090-4093-3/16/$31.00 ©2016 IEEE

stance of Rosenbrock function. Working with CC framework, gDG also shows better solution quality than DG and several state-of-the-art decomposition methods.

II. Related Work

In this section, we firstly describe the definition of separability and nonseparability. Then the studies of variable interaction learning is discussed briefly within CC framework context. Significantly, the main rule used in DG is described in details. In the continuous optimization literature, epistasis refers to nonseparability. The following rules can be used to determined whether a function is separable or nonseparable[1]:

Definition 1: A function $f(x)$ is separable iff

$$arg \min_{(x_1,\ldots,x_D)} f(x_1,\ldots,x_D) = \left(arg\min_{x_1} f(x_1,\ldots),\ldots,arg\min_{x_D} f(\ldots,x_D)\right). \quad (1)$$

Definition 2: A nonseparable function $f(x)$ is called m-nonseparable function if at most m of its parameters x_i are not independent. A nonseparable function $f(x)$ is called fully nonseparable function if any two of its parameters x_i are not independent.

In other words, variables in separable function can be allocated randomly in subcomponents, which can be optimized by some EAs within the context of CC framework. As for nonseparable functions, the values of the interacting variables should be considered when optimizing a variable x_i. According to the definition of separability, the difficulty of optimization problems can be estimated: the separable functions are the most easy ones whereas tackling fully nonseparable functions is the most challenging task.

In CC framework, decomposition methods can be designed through the heuristics of variable interaction. Random methods discover interdependencies between variables without prior knowledge. Random grouping [5] proposed by Yang et al. permutes the order of decision variables at the beginning of every co-evolutionary cycle, which increases the probability of putting two interacting variables in the same subcomponent. Nevertheless, the major drawback of this simple technique is the probability of allocating all interacting variables drops sharply when the number of them is greater than two [7].

Interaction adaptation methods such as LEGO [14], CCEA-AVP [15] uses the evolution of population to detect interaction. These methods rely on the relatively good individuals as they often reflect grouping behavior of interaction variables.

Perturbation methods tend to perturb the decision variables using various heuristics. Interactions between variables can be identified by observing the changes of objective function. In most cases, the decomposition stage is performed before co-evolution of subcomponents and consume some fitness evaluations. The optimization stage starts after the interaction structure is realized. Many perturbation techniques emerges in recent years and show their powerful decomposition results, such as CC with variable interaction learning(CCVIL) [16], cooperative particle swarm optimizer with statistical variable interdependence learning(CPSO-SL) [17]. Nevertheless, more remains to be done to provide these heuristics with theoretical basis.

DG is a perturbation method introduce by Omidvar et al.. The pairwise interaction is observed if (2) is satisfied.

$$\Delta_{\delta,x_i}[f]\vec{(x)}|_{x_i=a,x_j=b_1} \neq \Delta_{\delta,x_i}[f]\vec{(x)}|_{x_i=a,x_j=b_2}, \quad (2)$$

$$\Delta_{\delta,x_i}[f]\vec{(x)} = f(...,x_i+\delta,...) - f(...,x_i,...). \quad (3)$$

(2) shows that when the value of f is changed by add a perturbation to x_i for different values of x_j, x_i and x_j are interacting variables.

III. Graph-based Differential Grouping

In this section, the decomposition method is studies from the view of graph [18]. we show how to construct a graph for decision variables in a function and decompose them by finding connected components using Depth-first search algorithm [19]. The proposed method is called gDG as it applies the main rule in DG during the decomposition stage.

In mathematics, a graph is an ordered pair $G = (V, E)$ in which V represents a set of vertices and E represents a set of edges that are 2-elements subsets of V. A graph G is connected if . Otherwise G is disconnected. A connected component is a maximal connected subgraph of G.

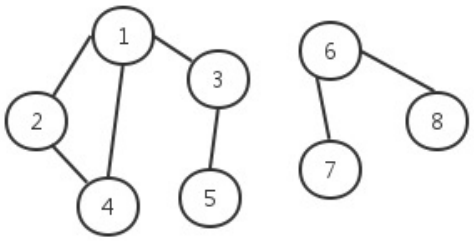

Fig. 1. A general graph

Fig. 1 is a graph with eight vertices and seven edges. It is also a disconnected graph with two connected components. In gDG, a graph is constructed for the objective function with decision variables being vertices. The weighted edge between two vertices is defined according to (4).

$$w_{ij} = \left|\Delta_{\delta,x_i}[f]\vec{(x)}|_{x_i=a,x_j=b_1} - \Delta_{\delta,x_i}[f]\vec{(x)}|_{x_i=a,x_j=b_2}\right|. \quad (4)$$

It is noteworthy that (4) is the transformation of (2) before being changed into absolute value. The value of w_{ij} reflect the strength of pairwise interaction, a higher value of w_{ij} indicates higher degree of interaction between two variables. Note that if there is no interaction between x_i and x_j, $w_{ij} = 0$. As for different functions, the scale of elements in w may be various, we use (5) to normalize w.

$$w_{ij} = \frac{w_{ij} - w_{min}}{w_{max} - w_{min}}. \quad (5)$$

w_{min} and w_{max} are the maximum and minimum element respectively in w. According to definitions of vertices and

Algorithm 1 $allgroups \leftarrow gDG\left(func, lb, ub, d, \sigma\right)$

Ensure: $d > 0 \wedge \sigma > 0$

1: $W \leftarrow zeros\left(d, d\right), dims \leftarrow \{1, 2, ..., d\}$
2: $seps \leftarrow \{\}, allgroups \leftarrow \{\}$
3: **for** $i = 1$ **to** d **do**
4: $\quad \overrightarrow{X_1} = lb \times ones\left(1, d\right)$
5: $\quad \overrightarrow{X_2} = \overrightarrow{X_1}$
6: $\quad \overrightarrow{X_2}\left(i\right) = ub$
7: $\quad \Delta_1 = func\left(\overrightarrow{X_1}\right) - func\left(\overrightarrow{X_2}\right)$
8: $\quad$ **for** $j = i + 1$ **to** d **do**
9: $\quad\quad \overrightarrow{X_1}\left(j\right) = (lb + ub)/2$
10: $\quad\quad \overrightarrow{X_2}\left(j\right) = (lb + ub)/2$
11: $\quad\quad \Delta_2 = func\left(\overrightarrow{X_1}\right) - func\left(\overrightarrow{X_2}\right)$
12: $\quad\quad w_{ij} = w_{ji} = |\Delta_1 - \Delta_2|$
13: $\quad$ **end for**
14: **end for**
15: **for** $i \in dims$ **do**
16: $\quad flag = 0$
17: $\quad$ **for** $j \in dims \wedge i \neq j$ **do**
18: $\quad\quad$ Normalize w_{ij} according to (5)
19: $\quad\quad$ **if** $w_{ij} < \sigma$ **then**
20: $\quad\quad\quad w_{ij} = 0$
21: $\quad\quad$ **else**
22: $\quad\quad\quad flag = 1$
23: $\quad\quad$ **end if**
24: $\quad$ **end for**
25: $\quad$ **if** $flag == 0$ **then**
26: $\quad\quad seps \leftarrow seps \cup \{i\}$
27: $\quad$ **end if**
28: **end for**
29: **if** !isempty($seps$) **then**
30: $\quad dims \leftarrow dims - seps$
31: $\quad allgroups = allgroups \cup \{seps\}$
32: **end if**
33: $(ng_1, ng_2, ..., ng_k) = graph\_con(W, dims, d)$
34: **for** $c = 1$ **to** k **do**
35: $\quad allgroups = allgroups \cup \{ng_c\}$
36: **end for**

Algorithm 2 $(ng_1, ng_2, ..., ng_k) = graph\_con(W, dims, d)$

global $count = 1$
global $id = -ones(1, d)$
global $visited = zeros(1, d)$
for $i \in dims$ **do**
$\quad$ **if** $!visited[i]$ **then**
$\quad\quad DFS(i)$
$\quad\quad count++$
$\quad$ **end if**
end for
for $c = 1$ **to** $count$ **do**
$\quad$ **for** $dim \in dims$ **do**
$\quad\quad$ **if** $id[dim] == c$ **then**
$\quad\quad\quad ng_c = ng_c \cup \{dim\}$
$\quad\quad$ **end if**
$\quad$ **end for**
end for

Algorithm 3 $DFS(i)$

$visited[i] = 1$
$id[i] = count$
for $j \in adj(i)$ **do**
$\quad$ **if** $!visited[j]$ **then**
$\quad\quad DFS(j)$
$\quad$ **end if**
end for

edges of an objective function, gDG is summarized as Algorithm 1.

In Algorithm1, there are five inputs: $func$ is the objective function; lb and ub are the lower bound and upper bound of each variable in $func$ respectively; d is the dimensionality of $func$; σ is the threshold to identify interaction between two variables.

gDG consists of three stages. Firstly, the strength of pairwise interaction is identified(Line 3-Line 14). Then gDG normalizes w and identify separable variables (Line 15-Line 28). The third stage is to construct a undirected weighted graph for interacting variables and form nonseparable subcomponents (Line 33). Finally, gDG returns all the subcomponents as the output (Line 34-Line 36).

The first stage begins with the calculation of the strength of pairwise interaction. This calculation can be conducted according to (4). The larger of w_{ij}, the stronger interaction between x_i and x_j. As is different from DG, gDG examined every pairwise interaction. Hence, the time complexity of this procedure is $O(D^2)$. gDG applies the same technique as DG to identify pairwise interaction[8](Line 3-Line 14). Different from DG, gDG does not use a threshold on W to identify whether pairwise interaction exists, but records the element in W directly (Line 12). In the second stage, (5) is applied to normalize w. x_i does not interact with x_j if $w_{ij} < \sigma$ for normalized w_{ij} and w_{ij} is set to zero (Line 19-Line 20). gDG uses $flag$ to indicate whether x_i is separable variable. For x_i, $flag$ is set to 0 if x_i is a separable variable (Line 25-Line 27), otherwise x_i interacts with at least one decision variable. All the separable variables form a group $seps$ if gDG identifies some separable variables. Decision variables in $seps$ are excluded from $dims$ as the graph is only constructed for interacting variables (Line 29-Line 32).

In the third stage, the algorithm tends to find the connected components of the graph constructed for nonseparable variables (Line 33). A connected component in a graph represents a group in which decision variables interact with each other. Fig. 2 shows a weighted graph constructed for nonseparable variables. In Fig. 2, x_1 interacts with x_2 and x_2 interacts with x_3, that is to say x_1 and x_2 as well as x_2 and x_3 should be put in the same group, which implies that x_1, x_2 and x_3 should be in the same group. According to the definition of

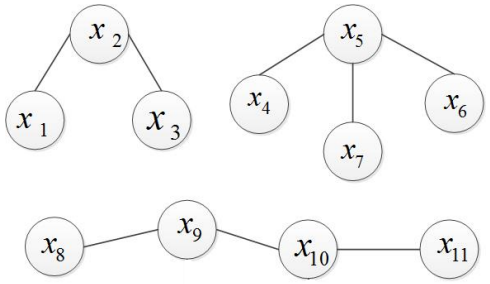

Fig. 2. Graph for interacting variables

connected component of a graph, any vertices in the same connected component are connected by paths, in which all vertices should be in the same group on the basis of the above derivation. Therefore, x_4-x_7 as well as x_8-x_{11} should be in the same group in Fig. 2. gDG uses the Depth-first search (DFS) (Algorithm 3) [19] to find the connected components in a graph and considers vertices in the same connected component as a nonseparable group (Algorithm 2). All the nonseparable groups are combined with the separable group to form the final output $allgroups$ (Line 41-Line 43).

IV. Experiments

The efficiency of gDG can be tested by a set of benchmark functions from CEC'2010. This set of benchmark functions were classified into five group in which objective functions have different properties. In this test suite, the dimensionality of each function is 1000.

A. Parameter Settings

We use SaNSDE [20], a variant of differential evolution [21], as the subcomponent optimizer in this paper. We conduct 25 independent runs for each objective function and calculate the mean and standard deviation of the optimizing results. The maximum number of function evaluations is 3e6, as suggested in [1]. In the second stage of gDG, the value of σ is set to 10^{-6} (Algorithm 1, Line 22), other value like 10^{-4}, 10^{-8} are used to test the sensitivity of gDG to σ .

V. Analysis of results

This section provides the analysis of decomposition results of gDG on 20 benchmark functions. We compare the grouping accuracy of gDG and DG in the decomposition stage. The sensitivity analysis of gDG on parameter σ is also presented in this section.

A. Performance of gDG

The performance of DG and gDG on 20 benchmark functions is showed in Table I. The first four columns are the properties of 20 benchmark functions. The first column is the function number while the second and third column are the separable variables and nonseparable variables respectively in each function. The forth column represents the number of nonseparable subcomponent of corresponding benchmark function. The last six column are decomposition results of corresponding decomposition method on benchmark functions. The ninth column records the number of fitness evaluation used in each decomposition method. The last column is the accuracy of each method when grouping interacting variables into the same subcomponent on each benchmark function. Note that the parameter σ is set to 10^{-6} in gDG and ε is set to 10^{-3} in DG due to the reason that these two methods perform best on the above parameter setting.

Table I shows that gDG outperforms DG on most benchmark functions. As suggested in [8], the grouping accuracy is reported as 100% if all interacting variables are allocated in the same subcomponent and independent variables are allocated in different subcomponent. Considering the grouping result of gDG on each function, the maximum numbers of interacting variables in all subcomponents are summed up to obtain the number of variables allocated correctly, which can be divided by the number of variables and obtain the grouping accuracy. Since there is no interacting variables in f_1, f_2 and f_3, the results of grouping accuracy are left blank on these functions. gDG achieves 100% on all benchmark functions except f_1-f_3 while DG achieves 100% on 10 benchmark functions. As for f_{13}, f_{18} and f_{20}, DG achieves very low grouping accuracy, but gDG achieves perfect performance on these three functions. Note that the number of fitness evaluation on each benchmark function used by gDG is 1001000, which is equal or greater than that used by DG. For each decision variable x_i, the number of fitness evaluation used to calculate Δ_1 is $2*D$. For each pairwise variables, the number of fitness evaluation used to calculate Δ_2 is $2 * C_D^2$. As $D = 1000$ in our experiment, the total number of fitness evaluation used in each benchmark function is equal to 1001000.

The first category of the benchmark functions are separable functions (f_1-f_3). DG identifies all the separable variables on these three functions successfully. gDG captures 24 and 238 separable variables on f_1 and f_2 respectively and puts all variables in one subcomponent on f_3 as nonseparable component. Since there is no interacting variables in f_1-f_3, the grouping accuracy of DG and gDG on these functions is left blank. The number of fitness evaluation used by gDG/DG is the same, which is 1001000/1001000.

There are five partially separable functions in the second category of the benchmark functions. Each of these functions contains a nonseparable subcomponent with 50 variables, the rest are separable variables. Note that gDG performs perfectly on all functions in category 2, which means that gDG identifies all the separable variables and interacting variables successfully on these functions. DG captures 33 separable variables on f_4 and forms 10 nonseparable subcomponents. As DG allocates all interacting variables in the same subcomponent, the grouping accuracy on f_4 is reported as 100%. DG achieves perfect decomposition on f_5 and f_6. As for f_7 and f_8, DG captures 248 and 134 separable variables, but misplaces 16 and 5 variables respectively. Therefore, the grouping accuracy of f_7 and f_8 are 68% and 90%. Consider the fitness evaluations used by gDG/DG on f_4, f_7 and f_8, gDG sacrifices much more fitness evaluations to achieve perfect decomposition results.

TABLE I
DECOMPOSITION RESULTS OF GRAPH BASED DIFFERENTIAL GROUPING AND DIFFERENTIAL GROUPING ON CEC'2010 BENCHMARK FUNCTIONS (SEPARATED BY "/")

Function	Sep Vars	Non-sep Vars	Non-sep Groups	Graph based Differential Grouping ($\sigma = 10^{-6}$) / Differential Grouping ($\varepsilon = 10^{-3}$)					
				#Captured Sep Vars	#Captured Non-sep Vars	#Formed Non-sep Groups	#Misplaced Vars	#FE	Grouping Accuracy
f_1	1000	0	0	24/1000	976/0	1/0	0/0	1001000/1001000	–
f_2	1000	0	0	238/1000	762/0	1/0	0/0	1001000/1001000	–
f_3	1000	0	0	0/1000	1000/0	1/0	0/0	1001000/1001000	–
f_4	950	50	1	950/33	50/50	1/10	0/0	1001000/14554	100%/100%
f_5	950	50	1	950/950	50/50	1/1	0/0	1001000/905450	100%/100%
f_6	950	50	1	950/950	50/50	1/1	0/0	1001000/906332	100%/100%
f_7	950	50	1	950/248	50/34	1/4	0/16	1001000/67742	100%/68%
f_8	950	50	1	950/134	50/45	1/5	0/5	1001000/23286	100%/90%
f_9	500	500	10	500/500	500/500	10/10	0/0	1001000/270802	100%/100%
f_{10}	500	500	10	500/500	500/500	10/10	0/0	1001000/272958	100%/100%
f_{11}	500	500	10	500/501	500/499	10/10	0/1	1001000/270640	100%/99.8%
f_{12}	500	500	10	500/500	500/500	10/10	0/0	1001000/271390	100%/100%
f_{13}	500	500	10	500/131	500/159	10/34	0/341	1001000/50328	100%/31.8%
f_{14}	0	1000	20	0/0	1000/1000	20/20	0/0	1001000/21000	100%/100%
f_{15}	0	1000	20	0/0	1000/1000	20/20	0/0	1001000/21000	100%/100%
f_{16}	0	1000	20	0/4	1000/996	20/20	0/4	1001000/21128	100%/99.6%
f_{17}	0	1000	20	0/0	1000/1000	20/20	0/0	1001000/21000	100%/100%
f_{18}	0	1000	20	0/78	1000/230	20/50	0/770	1001000/39624	100%/23%
f_{19}	0	1000	1	0/0	1000/1000	1/1	0/0	1001000/2000	100%/100%
f_{20}	0	1000	1	0/33	1000/287	1/241	0/713	1001000/155430	100%/28.7%

Category 3 consists 5 nonseparable functions (f_9-f_{13}) containing 10 nonseparable subcomponents, each with 50 interacting variables. The separable variables and interacting variables on these functions are 500. gDG and DG achieve perfect decomposition results on f_9, f_{10} and f_{12}. On f_{11}, gDG performs perfectly while DG identifies one more separable variable and misplaces one interacting variable. The grouping accuracy of gDG/DG is 100%/99.8%. DG can only identify 131 separable variables and misplaces 341 interacting variables on f_{13} which is the instance of Rosenbrock function. Nevertheless, gDG still obtains perfect decomposition result on f_{13}. The grouping accuracy of gDG on f_{13} is 100%, which is much higher than that obtained by DG. In Rosenbrock function, DG can identify x_i and x_{i+1} ($1 \leq i < D$) as interacting variables, but fails to identify x_i and x_j as interacting variables when $1 < |i - j| < D$. Therefore DG will break the nonseparable subcomponent into several nonseparable ones, leading the grouping accuracy on f_{13} to be low. As for gDG, it constructs a connected component for Rosenbrock function like x_8-x_{11} in Fig. 2, which identifies as a nonseparable subcomponent. Further more, gDG can identify interacting variables as long as they are in the same connected component.

Category 4 contains 5 nonseparable functions (f_{14}-f_{18}) containing 20 nonseparable subcomponents, each with 50 interacting variables. No separable variables are in these functions. gDG obtains perfect decomposition results on all functions, which means gDG identifies all separable variables and interacting variables correctly. DG performs perfectly on f_{14}, f_{15} and f_{17}, but misplaces 4 interacting variables on f_{16} and performs poorly on f_{18}, an instance of Rosenbrock function. Note that the number of fitness evaluation used by gDG is 1001000, which is much greater than that used by DG.

Category 5 contains 2 fully nonseparable functions (f_{19}-f_{20}). gDG achieves perfect decomposition results on f_{19} and f_{20} again. DG performs perfectly on f_{19} but performs poorly on Rosenbrock function f_{20}. Note that number of fitness evaluation used by DG on these two functions are much smaller than that used by gDG.

In summary, DG can obtain high grouping accuracy on most benchmark function, but performs poorly on the instances of Rosenbrock functions like f_{13}, f_{18} and f_{20}. gDG can identify all the separable variables and interacting variables on 20 benchmark functions. gDG can identify interaction variables as long as they are in the same connected component of the graph constructed for interacting variables. Meanwhile, gDG consumes more fitness evaluations to allocate all the variables correctly. This represents a classic trade-off between efficiency and accuracy.

The parameter σ is used as a threshold to identify whether interaction exists between two variables. Large σ leads gDG to consider more pairwise variables as independent. The smaller σ is, the more pairwise interaction gDG can identify. Meanwhile, the decomposition result of gDG can be affected by system errors if σ is too small. Too large σ will make gDG fail to identify interacting variables. Table II presents the decomposition results of gDG on 20 benchmark functions with $\sigma = 10^{-4}$ and $\sigma = 10^{-8}$.

Table I and Table II shows that the value of σ can influence the decomposition results of gDG on benchmark functions. gDG with $\sigma = 10^{-6}$ as well as $\sigma = 10^{-8}$ achieve 100% grouping accuracy on all benchmark functions except separable functions. Consider their decomposition results on f_{11}, gDG with $\sigma = 10^{-6}$ identifies all separable variables and interacting variables while gDG with $\sigma = 10^{-8}$ forms 11

TABLE II
DECOMPOSITION RESULTS OF GRAPH BASE DIFFERENTIAL GROUPING WITH DIFFERENT VALUES OF σ ON CEC'2010 BENCHMARK FUNCTIONS. THE RESULT OF $\sigma = 10^{-4}$ AND $\sigma = 10^{-8}$ ARE SEPARATED BY "/")

Function	Sep Vars	Non-sep Vars	Non-sep Groups	Graph based Differential Grouping ($\sigma = 10^{-4}$) / Graph based Differential Grouping ($\sigma = 10^{-8}$)					
				#Captured Sep Vars	#Captured Non-sep Vars	#Formed Non-sep Groups	#Misplaced Vars	#FE	Grouping Accuracy
f_1	1000	0	0	24/24	976/976	1/1	0/0	1001000/1001000	–
f_2	1000	0	0	238/238	762/762	1/1	0/0	1001000/1001000	–
f_3	1000	0	0	0/0	1000/1000	1/1	0/0	1001000/1001000	–
f_4	950	50	1	950/950	50/50	1/1	0/0	1001000/1001000	100%/100%
f_5	950	50	1	950/950	50/50	1/1	0/0	1001000/1001000	100%/100%
f_6	950	50	1	950/950	50/50	1/1	0/0	1001000/1001000	100%/100%
f_7	950	50	1	950/950	50/50	1/1	0/0	1001000/1001000	100%/100%
f_8	950	50	1	950/950	50/50	1/1	0/0	1001000/1001000	100%/100%
f_9	500	500	10	500/500	500/500	10/10	0/0	1001000/1001000	100%/100%
f_{10}	500	500	10	500/500	500/500	10/10	0/0	1001000/1001000	100%/100%
f_{11}	500	500	10	500/0	500/1000	10/11	0/0	1001000/1001000	100%/100%
f_{12}	500	500	10	500/500	500/500	10/10	0/0	1001000/1001000	100%/100%
f_{13}	500	500	10	500/500	500/500	10/10	0/0	1001000/1001000	100%/100%
f_{14}	0	1000	20	0/0	1000/1000	20/20	0/0	1001000/1001000	100%/100%
f_{15}	0	1000	20	0/0	1000/1000	20/20	0/0	1001000/1001000	100%/100%
f_{16}	0	1000	20	0/0	1000/1000	20/20	0/0	1001000/1001000	100%/100%
f_{17}	0	1000	20	0/0	1000/1000	20/20	0/0	1001000/1001000	100%/100%
f_{18}	0	1000	20	0/0	987/1000	22/20	13/0	1001000/1001000	98.7%/100%
f_{19}	0	1000	1	0/0	1000/1000	1/1	0/0	1001000/1001000	100%/100%
f_{20}	0	1000	1	0/0	614/1000	5/1	386/0	1001000/1001000	61.4%/100%

nonseparable groups as it allocates all separable variables in a nonseparable subcomponent. The reason may be that σ is too small so that the system errors may lead gDG to identify separable variables as interacting ones. As gDG obtains perfect decomposition results on all interacting variables, the grouping accuracy is reported as 100% on f_{11}.

gDG with $\varepsilon = 10^{-4}$ obtains 100% grouping accuracy on all benchmark functions except f_{18} and f_{20} which are instances of Rosenbrock function. The reason may be that the capability of gDG on handling Rosenbrock functions is affected by the value of σ. Therefore, gDG has to tune ε correctly when handling instance of Rosenbrock function. when $\sigma = 10^{-4}$, gDG break two nonseparable groups into four nonseparable groups on f_{18} and break the nonseparable group into five nonseparable groups on the fully nonseparable function f_{20}. The value of σ is too large so that gDG fails to identify the interaction when one variable has weak interaction with the other.

VI. DECC COMPARISION

This section combines our proposed decomposition method with DECC algorithm (DECC-gDG) and evaluates its performance on the benchmark of CEC'2010. The optimization results of DECC-gDG is compared with the baseline DECC-G[5], DECC-D[7] and DECC-DG[8].

Table III shows that DECC-D outperforms other methods on f_1 and f_3 significantly. DECC-G performs significantly better than other approaches on f_2. The reason may be that gDG forms two nonseparable subcomponents on f_1 and f_2 as well as only one subcomponent on f_3 while DG allocates all variables into one subcomponent on f_1, f_2 and f_3. When more variables are involved in fully separable functions, the solution quality will deteriorate if there are only a few subcomponents are formed [22] or treat each variable as one subcomponent [23]. Decomposition methods like DECC-D and DECC-G seem to be more efficient compared with these two extreme cases. Another reason may be that gDG and DG consumes more number of fitness evaluation and fewer number of fitness evaluation than DECC-D and DECC-G left for the optimization stage. Note that DECC-gDG and DECC-DG perform similarly on fully separable functions.

On partially nonseparable functions in category 2, gDG-DECC outperforms other methods significantly on f_4, f_7 and f_8. DECC-gDG and DECC-DG obtain similar results and perform better than DECC-D and DECC-G on f_5 and f_6. Note that gDG and DG get perfect decomposition results on f_5 and f_6 and their number of function evaluations used in the decomposition stage is rough the same. Therefore, DECC-gDG and DECC-DG performs similarly on these two functions. As for f_4, f_7 and f_8, gDG get perfect decomposition results while DG fails to capture some separable variables. The reason may be that identifying all separable variables and allocating them in the same subcomponent will be useful. Though gDG consumes much more function evaluations than DG, DECC-gDG outperforms DECC-DG on f_4, f_7 and f_8.

On partially nonseparable functions except f_{13} and f_{18} in category 3 and category 4, DECC-gDG achieves the same or slightly worse results than DECC-DG. The reason is that both gDG and DG have good performance on these functions, but gDG uses 5 times/50 times the number of function evaluations used by DG on f_9-f_{12}/f_{14}-f_{17}. Therefore, DECC-gDG has less number of fitness evaluation than DECC-DG, which results in better performance of DECC-DG. Consider function f_{13} and f_{18}, which are instances of Rosenbrock

TABLE III
COMPARISION OF DECC-GDG, DECC-DG, DECC-D AND DECC-G USING 25 INDEPENDENT RUNS ON BENCHMARK FUNCTIONS (WILCOXON TEST, $\alpha = 0.05$)

Functions		DECC-gDG	DECC-DG	DECC-D	DECC-G
f_1	mean	3.60e+05	1.16e+04	**3.20e-25**	8.68e-14
	std	3.18e+05	5.48e+04	1.58e-24	5.92e-14
f_2	mean	3.18e+03	4.48e+03	2.89e+02	**1.21e+02**
	std	1.24e+02	1.82e+02	2.37e+01	2.58e+01
f_3	mean	1.65e+01	1.67e+01	**1.24e-13**	1.93e+00
	std	3.34e-01	3.42e-01	5.20e-15	3.76e-01
f_4	mean	**1.03e+12**	4.54e+12	3.45e+12	1.21e+13
	std	4.16e+11	1.85e+12	1.12e+12	3.14e+12
f_5	mean	**1.49e+08**	**1.55e+08**	2.55e+08	2.57e+08
	std	2.36e+07	1.94e+07	5.90e+07	6.49e+07
f_6	mean	**1.63e+01**	**1.64e+01**	5.68e-09	5.25e+06
	std	2.81e-01	3.45e-01	1.78e-09	1.02e+06
f_7	mean	**8.82e+02**	1.20e+04	4.10e+08	4.41e+06
	std	1.02e+03	1.14e+04	2.63e+08	4.11e+06
f_8	mean	**4.79e+05**	3.50e+07	1.28e+08	7.51e+07
	std	1.32e+06	2.38e+07	1.75e+08	2.36e+07
f_9	mean	1.15e+08	**5.72e+07**	**6.04e+07**	2.37e+08
	std	1.41e+07	6.20e+06	8.02e+06	2.54e+07
f_{10}	mean	**5.30e+03**	**4.54e+03**	1.30e+04	9.46e+03
	std	1.67e+02	1.38e+02	2.65e+02	3.94e+02
f_{11}	mean	1.08e+01	1.02e+01	**4.11e-02**	2.57e+01
	std	1.05e+00	1.07e+00	2.05e-01	1.23e+00
f_{12}	mean	1.23e+04	**2.75e+03**	4.38e+06	4.53e+04
	std	1.23e+03	1.44e+03	1.73e+05	5.50e+03
f_{13}	mean	**2.96e+03**	4.22e+06	**1.30e+03**	3.90e+03
	std	9.39e+02	2.29e+06	4.56e+02	3.80e+03
f_{14}	mean	6.16e+08	3.43e+08	**2.00e+08**	6.10e+08
	std	4.35e+07	2.40e+07	1.61e+07	4.54e+07
f_{15}	mean	6.37e+03	**5.86e+03**	1.59e+04	7.57e+03
	std	9.61e+01	7.34e+01	3.24e+02	2.74e+03
f_{16}	mean	3.05e-08	**7.36e-13**	1.72e+01	8.42e+01
	std	2.82e-09	5.91e-14	8.54e+01	1.11e+01
f_{17}	mean	1.34e+05	**4.03e+04**	7.55e+06	1.76e+05
	std	5.56e+03	2.46e+03	4.09e+05	1.17e+04
f_{18}	mean	**1.39e+03**	1.10e+10	1.72e+03	1.69e+04
	std	1.62e+02	2.28e+09	4.38e+02	8.83e+03
f_{19}	mean	2.05e+06	1.75e+06	1.95e+07	**7.84e+05**
	std	1.05e+05	8.45e+04	1.77e+06	3.63e+04
f_{20}	mean	9.16e+04	7.39e+07	**1.14e+03**	3.44e+03
	std	9.76e+04	6.28e+07	8.85e+01	1.75e+02

function, gDG obtains perfect decomposition on these two functions. Although much less number of function evaluation is left for DECC-gDG in optimization stage, DECC-gDG still outperforms DECC-DG significantly on f_{13} and f_{18}.

On fully nonseparable function f_{19}, gDG and DG successfully place all interacting variables in the same subcomponent. On f_{20}, gDG identifies all interacting variables and DECC-gDG outperforms DECC-DG significantly. But the solution quality obtained by DECC-gDG and DECC-DG is worse than DECC-D and DECC-G. The reason for this may be that allocating all interacting variable in the same subcomponent is not a good choice for fully nonseparable functions.

In summary, DECC-gDG and DECC-DG perform better than DECC-D and DECC-G on most benchmark functions. DECC-gDG outperforms DECC-DG on all instances of Rosenbrock functions although more function evaluations are consumed by gDG in decomposition stage.

VII. CONCLUSION

In this paper, we have proposed graph-based differential grouping, an improved version of differential grouping, which decomposes an optimization problem into a set of subcomponents where there are few or no interdependency between them. We have shown how gDG applies graph theory to construct a modeling for the decomposition problem. For each graph, we use a threshold σ to identify separable variables and capture nonseparable subcomponents by finding connected components of the graph. Analysis on σ has been conducted to test the sensitivity of gDG. Experimental results have shown that gDG can identify all separable variables and nonseparable subcomponents on a suite of benchmark functions. Significantly, gDG is not sensitive to σ as it can get perfect grouping accuracy when σ is changed from 10^{-6} to 10^{-8}. When gDG was embedded with CC framework, it achieves better solution quality than DECC-DG, DECC-D and DECC-G on the instances of Rosenbrock function. gDG also obtains good performance on the other benchmark functions.

ACKNOWLEDGMENT

The authors would like to thank Xiaodong Li for providing source code of DECC-G, DECC-D and DECC-DG. This work is supported by the Key Laboratory of Machine Intelligence and Sensor Network, Ministry of Education, China under Grant No. 61379060, and by the National Natural Science Foundation of China under Grant No. 61303001, and by the Special Program for Applied Research on Super Computation of the NSFC-Guangdong Joint Fund (the second phase).

REFERENCES

[1] K. Tang, X. Yao, and P. Suganthan, "Benchmark functions for the CEC'2010 special session and competition on large scale global optimization," proceedings of Congress on Evolutionary Computation, Barcelona, Spain, July, 2010, pp. 1-23.

[2] M. Potter and K. D. Jong, "A cooperative coevolutionary approach to function optimization," proceedings of Parrallel Problem Solving from Nature, Jerusalem, Israel, October, 1994, pp. 249-257.

[3] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. USA: Addison-Wesley, 1989.

[4] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution," proceedings of IEEE Congress on Evolutionary Computation, Seoul, Korea, May, 2001, pp. 1101-1108.

[5] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," Inf Sci, vol. 178, no. 15, pp. 2985-2999, 2008.

[6] M. N. Omidvar, X. Li, Z. Yang, and X. Yao, "Cooperative co-evolution for large scale optimization through more frequent random grouping," proceedings of IEEE Congress on Evolutionary Computation, Barcelona, July, 2010, pp. 1754-1761.

[7] M. N. Omidvar, X. Li, and X. Yao, "Cooperative co-evolution with delta grouping for large scale non-separable function optimization," proceeding of IEEE Congress on Evolutionary Computation, Barcelona, July, 2010, pp. 1762-1769.

[8] M. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," IEEE Trans. Evol. Comput., vol. 18, no. 3, pp. 378-393, 2014.

[9] H. H. Rosenbrock, "An Automatic Method for Finding the Greatest or Least Value of a Function," Comput. J., vol. 3, no. 3, pp. 175-184, 1960.
[10] Y. Sun, M. Kirley, S. K. Halgamuge, "Extended Differential Grouping for Large Scale Global Optimization with Direct and Indirect Variable Interactions," proceedings of Conference on Genetic and Evolutionary Computation, Madrid, Spain, July, 2015.
[11] M. Ptashne, "How gene activators work," Sci. Amer., vol. 260, no. 1, pp. 40-47, 1989.
[12] T. Weise, R. Chiong, and K. Tang, "Evolutionary optimization: Pitfalls and booby traps," J. Comput. Sci. Technol., vol. 27, no. 5, pp. 90-936, 2012.
[13] D. Goldberg, B. Korb, and K. Deb, "Messy genetic algorithms: Motivation, analysis, and first results," Complex Syst., vol. 3, no. 5, pp. 493-530, 1989.
[14] J. Smith and T. C. Fogarty, Evolutionary Computing. London, U.K.: Springer-Verlag, 1995
[15] T. Ray and X. Yao, "A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning," proceeding of IEEE Congress on Evolutionary Computation, Trondheim, Norge, May, 2009, pp. 983-989.
[16] W. Chen, T. Weise, Z. Yang, and K. Tang, "Large-scale global optimization using cooperative coevolution with variable interaction learning," proceedings of International Conference on Parallel Problem Solving from Nature, Krakw, Poland, September, 2011, pp. 300-309.
[17] L. Sun, S. Yoshida, X. Cheng, Y. Liang, "A cooperative particle swarm optimizer with statistical variable interdependence learning," Inf. Sci., vol. 186, no. 1, pp. 20-39, 2012.
[18] F. Riaz and K. M. Ali, "Applications of Graph Theory in Computer Science," proceedings of International Conference on Computational Intelligence, Communication Systems and Networks, Bali, Indonesia, 2011, pp. 142-145.
[19] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to Algorithms 2nd Edition. USA.: MIT Press, 2001.
[20] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search, proceedings of IEEE Congress on Evolution Computation", Hong Kong, China, June, 2008, pp. 1110 - 1116.
[21] R. Storn and K. Price, "Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces," Global Optimiz., vol. 11, no. 4, pp. 341-359, 1995.
[22] Y. J. Shi, H. F. Teng, Z. Q. Li, "Cooperative Co-evolutionary Differential Evolution for Function Optimization," proceedings of International Conference on Advances in Natural Computation, Changsha, China, August, 2005, pp. 1080-1088.
[23] F. VandenBergh, A. Engelbrecht, "A Cooperative Approach to Particle Swarm Optimization," Evol. Comput., vol. 8, no. 3, pp. 225-239, 2004.