Extended Differential Grouping for Large Scale Global Optimization with Direct and Indirect Variable Interactions

Yuan Sun
Department of Mechnical Engineering
The University of Melbourne
Parkville, Australia
yuans2@student.unimelb.edu.au

Michael Kirley
Department of Computing and Information Systems
The University of Melbourne
Parkville, Australia
mkirley@unimelb.edu.au

Saman K. Halgamuge
Department of Mechnical Engineering
The University of Melbourne
Parkville, Australia
saman@unimelb.edu.au

ABSTRACT

Cooperative co-evolution is a framework that can be used to effectively solve large scale optimization problems. This approach employs a divide and conquer strategy, which decomposes the problem into sub-components that are optimized separately. However, solution quality relies heavily on the decomposition method used. Ideally, the interacting decision variables should be assigned to the same sub-component and the interdependency between sub-components should be kept to a minimum. Differential grouping, a recently proposed method, has high decomposition accuracy across a suite of benchmark functions. However, we show that differential grouping can only identify decision variables that interact directly. Subsequently, we propose an extension of differential grouping that is able to correctly identify decision variables that also interact indirectly. Empirical studies show that our extended differential grouping method achieves perfect decomposition on all of the benchmark functions investigated. Significantly, when our decomposition method is embedded in the cooperative co-evolution framework, it achieves comparable or better solution quality than the differential grouping method.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization

Keywords

Variable interaction, problem decomposition, cooperative co-evolution, large scale global optimization

1. INTRODUCTION

Evolutionary algorithms (EAs) are meta-heuristics that can be used to solve a wide range of optimization problems. However, when the problem has a large number of decision variables – *large scale global optimization* – it becomes difficult for an EA to find the optimal solution [8, 22].

Cooperative co-evolution (CC) [16] has been used with some successes when tackling large scale global optimization (eg. [12]). In CC, the optimization problem is divided into sub-components that are evolved independently. The final solution is a concatenation of representatives from each of the sub-components. In the original paper by Potter et al. [16], it was shown that the CC genetic algorithm was able to solve separable optimization problems, however it was ineffective on non-separable problems. These results may in part be attributed to the fact that the CC approach did not take variable interactions into consideration when allocating decision variables to sub-components.

When using a CC framework, it has been shown that the overall performance is correlated with the decomposition method used [1, 12]. Ideally, the interacting decision variables should be assigned to the same sub-component and the interdependence between sub-components should be kept to a minimum. Unfortunately, identifying the interacting variables in many optimization problems is non-trivial.

Recently, Omidvar et al. [12] proposed a decomposition method – *differential grouping* (DG) – that can be used when allocating decision variables to sub-components when using the CC framework. Results from detailed numerical simulation experiments indicate that the DG method had high decomposition accuracy on most of the benchmark functions investigated [20]. However, on some benchmark functions, such as the Rosenbrock functions, the decomposition accuracy was very low. This suggests that the decomposition method described did not cover all forms of decision variable interactions.

In this paper, we investigate the effects of interactions between decision variables and decomposition methods within the CC framework. We start by considering the form of decision variable interaction in optimization problems such as the Rosenbrock functions. We suggest that there are two distinct types of variable interactions as shown in Figure 1. In Type I interactions, the variables interact directly eg. x_1 and x_2 (or x_2 and x_3) interact directly. In Type II interactions, the variables have a form of indirect interaction eg. x_1 and x_3 are linked by x_2. The formal definition of these two types of variable interactions will be given in Section 3. We show that DG can only capture Type I interactions and fails to capture Type II interactions between decision

GECCO '15, July 11 - 15, 2015, Madrid, Spain

DOI: http://dx.doi.org/10.1145/2739480.2754666

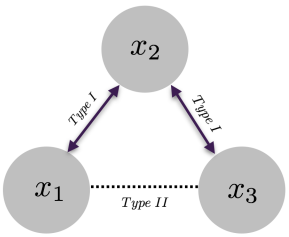

Figure 1: Two types of decision variable interaction. Type I: two variables interact directly with each other. Type II: two variables interact indirectly, that is they are linked via a third variable.

variables. We hypothesize that this is the reason why DG performs poorly on the Rosenbrock functions.

We propose an eXtended Differential Grouping (XDG) method to cater for the different forms of variable interactions. In XDG, after allocating decision variables that interact directly to nominated sub-components, "overlaps" between sub-components are identified. That is, when an overlap is observed, the sub-components that contain the same decision variables are merged. This searching–merging technique is employed to capture Type II interaction between decision variables. The efficacy of the XDG is evaluated using the benchmark functions from the CEC'2010 special session on large scale global optimization [20]. The experimental results show that XDG achieves perfect decomposition on all of the benchmark functions. We then embed XDG into the CC framework to solve large scale optimization problems. Empirical studies show that on the benchmark functions with Type II interactions, the proposed approach achieves significantly better results than the CC with standard DG. On the benchmark functions without Type II interactions, the proposed approach achieves comparable results with the CC with standard DG.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 shows the limitation of DG and proposes an extension to address this limitation. Section 4 sets up the experiments and analyses the experimental results. Section 5 concludes the paper.

2. RELATED WORK

In this section, we briefly describe studies related to variable interaction within an evolutionary computation context. We also summarize well-known decomposition methods used within the CC framework, before describing the DG method in detail.

Investigating the effects of interactions between "bits" or decision variables – epistasis or linkage – in genetic algorithm research has a long history (see [4]). The original motivation behind much of this work, was to improve the design and effectiveness of the genetic operators. For example, specific crossover operators allow a set of interacting binary variables to be inherited together in the evolutionary process, such as the model used in the messy GA [3], fast messy GA [5], gene expression messy GA [7]. Significantly, there is work describing the level of epistasis and problem difficulty. For example, papers describing epistasis variance [2], epistasis correlation [17] and entropic epistasis [18] have been used to quantify the variable interactions and problem difficulty.

In the continuous optimization domain, many studies have examined the effects of variable interactions and used this to guide the evolution trajectory. For example, estimation of distribution algorithms [9] build a probabilistic model based on the promising candidate solutions and updates the model during each generation. The population for the next generation is sampled from the probabilistic model. In the covariance matrix adaptation - evolutionary strategy [6], the population evolves with the mean and covariance matrix of the promising candidate solutions. In each generation, the mean and covariance matrix are calculated from the promising candidate solutions. The population for the next generation is sampled from the mean and covariance matrix.

In the CC framework, the decomposition methods that are typically used are based on some measures of decision variable interaction. Perhaps the simplest decomposition method is random grouping [23]. In the random grouping decomposition method, decision variable allocations are exchanged (or modified) a few times to increase the probability of assigning interacting decision variables into the same sub-component. However, it has been shown that when the number of interacting variables is greater than two, it is unlikely to put all of them into the same sub-component [13].

Perturbation methods, which include LINC-R [10], LIMD [11], CCVIL [1], differential grouping [12], and delta grouping [14] can also be used when allocating decision variables to sub-components. Perturbation methods identify interaction between two decision variables by adding a small perturbation to the decision variables and detecting the changes in the fitness values. However, these methods are sensitive to the computational errors in the system.

In the DG method introduced by Omidvar et al. [12], interactions between decision variables are identified according to the following rule: if

$$\Delta_{x_i} f(\vec{X})|_{x_i=a,x_j=b_1} \neq \Delta_{x_i} f(\vec{X})|_{x_i=a,x_j=b_2}, \qquad (1)$$

then x_i and x_j are interacting decision variables, where

$$\Delta_{x_i} f(\vec{X}) = f(\ldots, x_i + \delta, \ldots) - f(\ldots, x_i, \ldots). \qquad (2)$$

In DG, when changes in f caused by adding a perturbation to x_i varies for different values of x_j, then x_i and x_j are interacting decision variables.

3. EXTENDED DIFFERENTIAL GROUPING

In this section, we define two types of interactions between decision variables and show that the DG method can only identify one type of interaction. An extension of DG, which we refer to as XDG, is proposed to address this limitation.

In Figure 1, we illustrate two alternative forms of variable interactions between decision variables. In Type I, two variables interact directly with each other. In Type II, two variables are linked via a third decision variable. We call the former *direct interaction* and the latter *indirect interaction*. The formal definition of interacting types is listed below:

DEFINITION 1. *In an objective function $f(\vec{X})$, decision variables x_i and x_j interact directly with each other if $\exists$ a candidate solution $\vec{x}_*$, s.t.*

$$\frac{\partial f}{\partial x_i \partial x_j}\Big|_{\vec{x}_*} \neq 0, \qquad (3)$$

denoted by $x_i \leftrightarrow x_j$. Decision variables x_i and x_j interact indirectly with each other if for all candidate solutions,

$$\frac{\partial f}{\partial x_i \partial x_j} = 0, \tag{4}$$

and $\exists$ a set of decision variables $\{x_{k1}, \ldots, x_{kt}\} \subset \vec{X}$, s.t. $x_i \leftrightarrow x_{k1} \leftrightarrow \ldots \leftrightarrow x_{kt} \leftrightarrow x_j$. Decision variables x_i and x_j are independent with each other if for all candidate solutions, (4) holds and $\nexists$ a set of decision variables $\{x_{k1}, \ldots, x_{kt}\} \subset \vec{X}$, s.t. $x_i \leftrightarrow x_{k1} \leftrightarrow \ldots \leftrightarrow x_{kt} \leftrightarrow x_j$.

Consider the following example to further explain this definition:

EXAMPLE 1. *In the objective function: $f(\vec{X}) = (x_1 - x_2)^2 + (x_2 - x_3)^2 + x_4^2$, $\vec{X} \in [-1, 1]^4$, x_1 and x_2 interact directly with each other (Type I), x_1 and x_3 interact indirectly with each other (Type II), x_1 and x_4 are independent.*

THEOREM 1. *In an objective function $f(\vec{X})$, if (4) holds, then*

$$\Delta_{x_i} f(\vec{X})|_{x_i=a, x_j=b_1} = \Delta_{x_i} f(\vec{X})|_{x_i=a, x_j=b_2}, \tag{5}$$

where $\Delta_{x_i} f(\vec{X})$ is defined in (2).

PROOF. If (4) holds, integrating (4) with respect to x_j, we can obtain:

$$\frac{\partial f}{\partial x_i} = \int 0 \mathrm{d}x_j = g(\vec{X_k}), \tag{6}$$

where $\vec{X_k} \subset \vec{X}$, $x_j \notin \vec{X_k}$, $g(\vec{X_k})$ is a function of $\vec{X_k}$. That states $\partial f / \partial x_i$ is not a function of x_j. Therefore $\forall\ x_i$,

$$\frac{\partial f}{\partial x_i}\Big|_{x_j=b_1} = \frac{\partial f}{\partial x_i}\Big|_{x_j=b_2}. \tag{7}$$

That is

$$\frac{\Delta_{x_i} f(\vec{X})}{\delta}\Big|_{x_j=b_1} = \frac{\Delta_{x_i} f(\vec{X})}{\delta}\Big|_{x_j=b_2}. \tag{8}$$

Thus,

$$\Delta_{x_i} f(\vec{X})|_{x_j=b_1} = \Delta_{x_i} f(\vec{X})|_{x_j=b_2} \tag{9}$$

is true for all x_i. Thus, (5) is obtained. □

In an objective function, if x_i and x_j interact indirectly with each other, then (4) holds. According to Theorem 1, (5) holds. DG will classify x_i and x_j as independent. Therefore, DG can not capture indirect interaction (Type II) between decision variables. This is the main limitation of DG.

Now consider a second example:

EXAMPLE 2. *In the objective function: $f(\vec{X}) = (x_1 - x_2)^2 + (x_2 - x_3)^2$, $\vec{X} \in [-1, 1]^3$, x_1 and x_3 interact indirectly with each other (Type II). However, DG classifies x_1 and x_3 as independent.*

PROOF. On the one hand,

$$\begin{aligned}\Delta_{x_1} f(\vec{X})|_{x_1=a, x_3=b_1} &= (a+\delta-x_2)^2 + (x_2-b_1)^2 \\ &-(a-x_2)^2 - (x_2-b_1)^2 = (a+\delta-x_2)^2 - (a-x_2)^2.\end{aligned} \tag{10}$$

On the other hand,

$$\begin{aligned}\Delta_{x_1} f(\vec{X})|_{x_1=a, x_3=b_2} &= (a+\delta-x_2)^2 + (x_2-b_2)^2 \\ &-(a-x_2)^2 - (x_2-b_2)^2 = (a+\delta-x_2)^2 - (a-x_2)^2.\end{aligned} \tag{11}$$

Algorithm 1 XDG

Require: f, d, $\vec{ub}$, $\vec{lb}$, ϵ
1: $IM \leftarrow zeros(d-1, d)$ // IM: Interaction Matrix
2: $sep\_var \leftarrow []$ //sep_var: separable variables
3: **for** $i = 1$ **to** d **do**
4: $groups(i) = \{i\}$
5: $\vec{X}_1 \leftarrow \vec{lb}$
6: $\vec{X}_2 \leftarrow \vec{X}_1$
7: $\vec{X}_2(i) \leftarrow \vec{ub}(i)$
8: $\Delta_1 \leftarrow f(\vec{X}_1) - f(\vec{X}_2)$
9: **for** $j = i + 1$ **to** d **do**
10: **if** $IM(i, j) = 0$ **then**
11: $\vec{X}_1(j) \leftarrow (\vec{ub}(j) + \vec{lb}(j))/2$
12: $\vec{X}_2(j) \leftarrow (\vec{ub}(j) + \vec{lb}(j))/2$
13: $\Delta_2 \leftarrow f(\vec{X}_1) - f(\vec{X}_2)$
14: **if** $|\Delta_1 - \Delta_2| > \epsilon$ **then**
15: $IM(i, j) \leftarrow 1$
16: $groups(i) \leftarrow groups(i) \cup \{j\}$
17: **end if**
18: **else**
19: $groups(i) \leftarrow groups(i) \cup \{j\}$
20: **end if**
21: **end for**
22: **for** $p, q \in groups(i)$ & $p < q$ **do**
23: $IM(p, q) \leftarrow 1$
24: **end for**
25: **end for**
26: **while** the number of variables in $groups$ is not d **do**
27: **for** $p, q \in \{1 : num(groups)\}$ & $p < q$ **do**
28: **if** $groups(p) \cap groups(q) \neq \emptyset$ **then**
29: $groups(p) \leftarrow groups(p) \cup groups(q)$
30: delete $groups(q)$
31: **end if**
32: **end for**
33: **end while**
34: **for** $i = 1$ **to** num($groups$) **do**
35: **if** length($groups(i)$) $= 1$ **then**
36: $sep\_var \leftarrow sep\_var \cup group(i)$
37: delete $groups(i)$
38: **end if**
39: **end for**
40: $groups \leftarrow groups \cup \{sep\_var\}$
41: **return** $groups$

Therefore,

$$\Delta_{x_1} f(\vec{X})|_{x_1=a, x_3=b_1} = \Delta_{x_1} f(\vec{X})|_{x_1=a, x_3=b_2}. \tag{12}$$

DG classifies x_1 and x_3 as independent. □

Can this limitation be addressed? Definition 1 states that under the condition (4), if a set of decision variables can be found to link x_i and x_j together, x_i and x_j interact indirectly with each other. Inspired by this, we propose the XDG method (see in Algorithm 1) to address this limitation.

In Algorithm 1, the inputs are: f is the objective function; d is the dimensionality; $\vec{ub}$ and $\vec{lb}$ are the upper bounds and lower bounds of decision variables; ϵ is the threshold to identify direct interaction between decision variables.

There are three main stages in the XDG method. The first stage is to identify direct interaction between decision variables (line 3 to line 25). The second stage is to identify

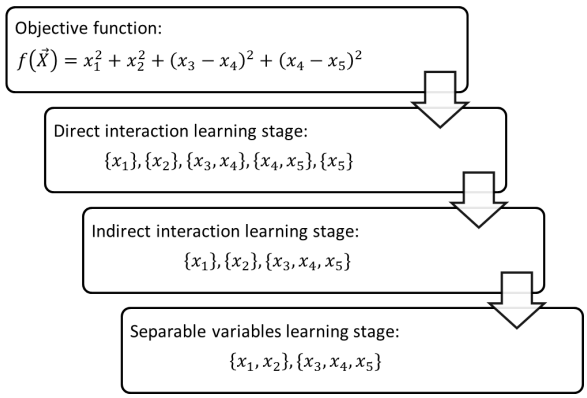

Figure 2: The decomposition processes used in XDG for a sample objective function. In the direct interaction learning stage there are 5 sub-components. In the indirect interaction learning stage the number of sub-components is reduced to 3. The separable variable learning stage results in 2 sub-components.

indirect interaction between decision variables (line 26 to line 33). The third stage is to group all of separable decision variables into the same sub-component (line 34 to line 40).

The first stage begins with the algorithm identifying all of the decision variables that interact directly with x_1, and placing them in the first sub-component with x_1. Note that the pairwise decision variables in the same sub-component are not needed to be examined again in the following procedure (line 22 to line 24). For example, if x_1 directly interacts with x_3 and x_4, we already know x_3 interacts with x_4, therefore, they do not need to be examined again in the following procedures. Secondly, the algorithm examines the direct interaction between x_2 and all of the other decision variables except x_1, for x_1 and x_2 have been examined in the previous procedure. This process is conducted for all decision variables and d sub-components are formed.

The technique used to identify direct interaction is the same as in the original DG method (line 5 to line 8, line 11 to line 17). To examine direct interaction between x_i and x_j, all decision variables are initialized to the lower bound of the search space in the vector of $\vec{X}_1$ (line 5). $\vec{X}_2$ is the same with $\vec{X}_1$ except for the i_{th} value, which is set to the upper bound of the search space (line 6 and line 7). The algorithm calculates the difference between the fitness values at $\vec{X}_1$ and $\vec{X}_2$, denoted by Δ_1 (line 8). Then the j_{th} value of $\vec{X}_1$ and $\vec{X}_2$ are set to the center of the search space. And the algorithm calculates the difference between the fitness values at $\vec{X}_1$ and $\vec{X}_2$ again, denoted by Δ_2 (line 11 to line 13). If the difference between Δ_1 and Δ_2 is greater than the threshold ϵ, x_i and x_j are classified as direct interaction.

In the second stage, the algorithm searches for overlaps between the d sub-components formed in the first stage. If the same decision variable appears in two sub-components, the algorithm merges the two sub-components (line 28 to line 31). This process is repeated until all sub-components are disjoint. Note that when all sub-components are disjoint, the total number of variables in all sub-components is equal to d. This is employed as the stopping criterion of the second stage (line 26).

In the third stage, the algorithm counts the number of decision variables in each sub-component. If it is equal to 1, the algorithm places the decision variable into the separable variable sub-component. Note that placing all separable variables into the same sub-component is arbitrary, which may not be the best decomposition method. However, we follow the approach used in DG to group separable variables in this way. The algorithm returns all of the sub-components as the output.

To further expand on the processing stages, consider Example 3 and Figure 2.

EXAMPLE 3. *Take the objective function:* $f(\vec{X}) = x_1^2 + x_2^2 + (x_3 - x_4)^2 + (x_4 - x_5)^2$, $\vec{X} \in [-1, 1]^5$ *as an example. The three stages of XDG on the objective function is illustrated in Figure 2.*

4. EXPERIMENTS

In this section, detailed numerical experiments are conducted to investigate the efficacy of XDG. Two research questions guide the experimental design:

Q1. Can the proposed XDG method address the limitation of the DG method identified in Section 3?

Q2. Can the proposed XDG method outperform the DG method when it is embedded in a CC framework for large scale optimization problems?

4.1 Methodology

To answer the two research questions, benchmark functions from the CEC'2010 special session on large scale global optimization [20] are used. This benchmark suite was also used to evaluate the DG method in [12]. The suite consists of 20 benchmark functions with 5 categories:

1. fully separable functions (f_1 to f_3);
2. partially separable functions with 1 non-separable sub-component (f_4 to f_8);
3. partially separable functions with 10 non-separable sub-components (f_9 to f_{13});
4. partially separable functions with 20 non-separable sub-components (f_{14} to f_{18});
5. fully non-separable functions (f_{19} to f_{20}).

The dimensionality of the 20 benchmark functions is $d = 1000$, and the number of decision variables in each sub-component is 50.

To investigate Q1, our XDG method is tested on the 20 benchmark functions. The threshold ϵ was set to 10^{-1}. (If not specified $\epsilon = 10^{-1}$ in the rest of the paper). Other values of ϵ (10^{-3} and 1) were used to test the sensitivity of XDG. Two metrics were employed to evaluate the performance of XDG: one is the number of function evaluations used to decompose the problem; the other is the accuracy of grouping of the interacting variables into the same sub-component. The performance of XDG is compared with the performance of DG. The value 10^{-3} is selected as the threshold ϵ for DG, as suggested in [12]. If not specified, all the parameter selections are consistent with the original paper. Note that the parameter settings for XDG and DG are different. For DG, $\epsilon = 10^{-3}$ performs better than $\epsilon = 10^{-1}$, as

Table 1: Decomposition results of XDG and DG on the 20 large scale benchmark functions. XDG and DG values are separated by "/".

Function	Sep Vars	Non-sep Vars	Non-Sep Groups	Extended Differential Grouping ($\epsilon = 10^{-1}$) / Differential Grouping ($\epsilon = 10^{-3}$)					
				Captured Sep Vars	Captured Non-sep Vars	Formed Non-sep Groups	Misplaced Vars	Function Evaluations	Grouping Accuracy
f_1	1000	0	0	1000/1000	0/0	0/0	0/0	1001000/1001000	100%/100%
f_2	1000	0	0	1000/1000	0/0	0/0	0/0	1001000/1001000	100%/100%
f_3	1000	0	0	1000/1000	0/0	0/0	0/0	1001000/1001000	100%/100%
f_4	950	50	1	0/33	50/50	1/10	0/0	80526/14554	100%/100%
f_5	950	50	1	950/950	50/50	1/1	0/0	998648/905450	100%/100%
f_6	950	50	1	950/950	50/50	1/1	0/0	998648/906332	100%/100%
f_7	950	50	1	950/248	50/34	1/4	0/16	998648/67742	100%/68%
f_8	950	50	1	0/134	50/45	2/5	0/5	121658/23286	100%/90%
f_9	500	500	10	500/500	500/500	10/10	0/0	977480/270802	100%/100%
f_{10}	500	500	10	500/500	500/500	10/10	0/0	977480/272958	100%/100%
f_{11}	500	500	10	500/501	500/499	10/10	0/1	978528/270640	100%/99.8%
f_{12}	500	500	10	500/500	500/500	10/10	0/0	977480/271390	100%/100%
f_{13}	500	500	10	500/131	500/159	10/34	0/341	1000154/50328	100%/31.8%
f_{14}	0	1000	20	0/0	1000/1000	20/20	0/0	953960/21000	100%/100%
f_{15}	0	1000	20	0/0	1000/1000	20/20	0/0	953962/21000	100%/100%
f_{16}	0	1000	20	0/4	1000/996	20/20	0/4	956286/21128	100%/99.6%
f_{17}	0	1000	20	0/0	1000/1000	20/20	0/0	953960/21000	100%/100%
f_{18}	0	1000	20	0/78	1000/230	20/50	0/770	999340/39624	100%/23%
f_{19}	0	1000	1	0/0	1000/1000	1/1	0/0	3998/2000	100%/100%
f_{20}	0	1000	1	0/33	1000/287	1/241	0/713	1001000/155430	100%/28.7%

shown in [12]. This is the reason why we select 10^{-3} instead of 10^{-1} as the ϵ value for DG.

To investigate Q2, we use the DECC cooperative coevolution algorithm/framework used by Omidvar et al. [12]. DECC uses SaNSDE [24] to optimize each sub-component. The population size was set to 50. The maximal number of function evaluations is set to 3×10^6, divided between the decomposition phase and the evolutionary optimization phase. The performance of XDG was compared with DG and delta grouping [14], where each decomposition method was incorporated into the DECC. These three algorithms are denoted by DECC-XDG (DECC with XDG), DECC-DG (DECC with DG) and DECC-D (DECC with delta grouping). For each algorithm, 25 independent runs are conducted for each benchmark function. The mean and standard deviation of the best solutions found in the fixed number of function evaluations are recorded to evaluate the performance of the algorithms. The two-sided Wilcoxon test with the confidence interval of 95% is used to determine the best performance from the 3 algorithms in a pairwise fashion.

4.2 Performance of XDG

In this section, we analyse the decomposition results of XDG on 20 benchmark functions. We compare the results from XDG with the decomposition results of DG. We also analyse the sensitivity of XDG to the parameter ϵ.

The decomposition results of XDG and DG on the 20 benchmark functions are presented in Table 1, which are separated by "/". The different categories of benchmark functions are separated by the double lines. The first 4 columns are the details for each benchmark function, and the last 6 columns are the results of the decomposition methods on each benchmark function. Specifically, the ninth column records the number of function evaluations used by XDG/DG to decompose each benchmark function. The last column records the accuracy of XDG/DG when grouping interacting variables into the same sub-component on each benchmark function.

Table 1 shows that XDG outperforms DG. In terms of grouping accuracy, XDG achieves 100% accuracy on all of the 20 benchmark functions. DG achieves 100% accuracy only on 13 benchmark functions. On functions f_{18} and f_{20}, the grouping accuracy of DG is very low (less than 30%). Note that the number of function evaluations used by XDG is equal or larger than the number used by DG. The extra function evaluations are used to identify indirect interaction between decision variables.

The first category of the 20 benchmark functions (See Section 4.1) contains three fully separable functions (f_1 to f_3). Both XDG and DG successfully identify all of the 1000 decision variables as separable. Note that the function evaluations used by XDG/DG on the three separable functions are the same, which is 1001000/1001000.

Category 2 consists of 5 partially separable functions (f_4 to f_8). Each function contains one non-separable group with 50 decision variables. On function f_5 f_6 and f_7, XDG successfully identifies all of the 50 interacting variables and 950 separable variables. On function f_4, XDG identifies all of the 1000 decision variables as interacting variables. It may seem odd that we still report the grouping accuracy as 100%. The reason is that XDG (or DG) only focuses on grouping interacting variables. As long as all of the interacting variables are grouped into the same sub-component, the grouping accuracy is 100%. This criterion is suggested in [12]. On function f_8, XDG forms two non-separable groups, one with 50 interacting variables and one with 950 separable variables. Note that XDG (or DG) automatically places all of the separable variables into the same sub-component. Therefore, the unexpected grouping of 950 separable variables will not affect the final decomposition result on f_8. DG also achieves perfect decomposition on f_5 and f_6. On f_7, the number of interacting variables identified by DG is 34 out of 50. Therefore the grouping accuracy of DG on f_7 is 68%. On f_8, the grouping accuracy of DG is 90%, with 45 out of 50 interacting variables successfully identified.

Category 3 functions (f_9 to f_{13}) contain 10 non-separable sub-components, each with 50 interacting decision variables. The number of separable and non-separable decision variables are both 500 in each function. On all of the 5 functions, XDG successfully identifies the 500 separable decision vari-

Table 2: Decomposition results of XDG with different values of threshold ϵ on 20 benchmark functions. The results of $\epsilon = 1$ and the results of $\epsilon = 10^{-3}$ are separated by "/".

Function	Sep Vars	Non-sep Vars	Non-Sep Groups	Extended Differential Grouping ($\epsilon = 1$)/ Extended Differential Grouping ($\epsilon = 10^{-3}$)					
				Captured Sep Vars	Captured Non-sep Vars	Formed Non-sep Groups	Misplaced Vars	Function Evaluation	Grouping Accuracy
f_1	1000	0	0	1000/1000	0/0	0/0	0/0	1001000/1001000	100%/100%
f_2	1000	0	0	1000/1000	0/0	0/0	0/0	1001000/1001000	100%/100%
f_3	1000	0	0	1000/1000	0/0	0/0	0/0	1001000/1001000	100%/100%
f_4	950	50	1	0/0	50/50	1/1	0/0	80526/80526	100%/100%
f_5	950	50	1	950/950	50/50	1/1	0/0	998648/998648	100%/100%
f_6	950	50	1	950/950	50/50	1/1	0/0	998648/998648	100%/100%
f_7	950	50	1	950/0	50/50	1/1	0/0	998648/45290	100%/100%
f_8	950	50	1	0/0	50/50	2/2	0/0	121658/1216586	100%/100%
f_9	500	500	10	500/500	500/500	10/10	0/0	977480/977480	100%/100%
f_{10}	500	500	10	500/500	500/500	10/10	0/0	977486/ 977480	100%/100%
f_{11}	500	500	10	1000/500	0/500	0/10	500/0	1001000/977482	0%/100%
f_{12}	500	500	10	500/500	500/500	10/10	0/0	977480/977480	100%/100%
f_{13}	500	500	10	500/4	500/50	10/2	0/450	1000154/ 573776	100%/10%
f_{14}	0	1000	20	0/0	1000/1000	20/20	0/0	953960/953960	100%/100%
f_{15}	0	1000	20	0/0	1000/1000	20/20	0/0	953978/ 953960	100%/100%
f_{16}	0	1000	20	1000/0	0/1000	0/20	1000/0	1001000/953968	0%/100%
f_{17}	0	1000	20	0/0	1000/1000	20/20	0/0	953960/ 953960	100%/100%
f_{18}	0	1000	20	0/0	1000/50	20/1	0/950	999340/ 171370	100%/5%
f_{19}	0	1000	1	0/0	1000/1000	1/1	0/0	3996/3996	100%/100%
f_{20}	0	1000	1	0/0	1000/1000	1/1	0/0	1001000/ 705698	100%/100%

ables and 10 non-separable groups, each with 50 interacting variables. DG also achieves perfect decomposition on f_9, f_{10} and f_{12}. On f_{11}, DG only misplaces 1 non-separable decision variable. On f_{13}, DG forms 34 non-separable groups. The number of interacting decision variables correctly identified is 159. Note that f_{13} is a Rosenbrock function, which contains examples of indirect interaction (Type II). It also shows that DG can not capture indirect interaction between decision variables. For this reason, DG may break one non-separable group into several groups. For example, in the function: $f(\vec{X}) = (x_1 - x_2)^2 + (x_2 - x_3)^2 + (x_3 - x_4)^2$, there is only one non-separable group: $\{x_1, x_2, x_3, x_4\}$. However DG will form two non-separable groups: $\{x_1, x_2\}$ and $\{x_3, x_4\}$. This is why DG forms 34 non-separable groups on f_{13}.

Category 4 functions (f_{14} to f_{18}) contain 20 non-separable groups, each with 50 interacting decision variables. XDG achieves perfect decomposition on all of the 5 functions. DG achieves 100% grouping accuracy on f_{14} f_{15} and f_{17}. On f_{16}, DG unexpectedly classifies 4 interacting variables as separable. Function f_{18} is another function with indirect interaction. DG forms 50 non-separable groups. The number of interacting decision variables correctly identified is 230. Therefore, the grouping accuracy of DG on f_{18} is 23%, which is very low. Note that the function evaluations used by XDG on all of the 5 functions are around one million, which is much greater than the function evaluations used by DG (around twenty thousand).

Category 5 consists of two fully non-separable functions with 1000 interacting decision variables (f_{19} and f_{20}). XDG successfully assigns all of the 1000 interacting variables into the same sub-component. DG also achieves 100% grouping accuracy on f_{19}. However on f_{20}, DG only correctly identifies 287 interacting decision variables. Note that f_{20} is also a Rosenbrock function. The number of function evaluations used by XDG and DG on f_{19} are both small. However on f_{20}, the number of function evaluations used by XDG is about 6 times greater than that used by DG.

In sum, DG achieves high grouping accuracy on most of the benchmark functions. However, it performs poorly on functions with indirect interaction, such as f_7 f_{13} f_{18} and f_{20}. XDG achieves 100% grouping accuracy on all of the 20 benchmark functions. It can successfully identify both direct interaction and indirect interaction. Therefore, the extension in XDG can address the limitation of the standard DG. However, XDG is more computational expensive than DG as it requires computational resources to identify indirect interaction between decision variables. This represents a classic trade-off between accuracy and efficiency.

The parameter ϵ is the threshold to classify pairwise decision variables as direct interaction or independent. The larger the ϵ is, the more likely the interacting decision variables are classified as independent. However, if the ϵ is too small, the independent decision variables may be classified as interacting due to the computational errors in the system. Table 2 presents the grouping results of XDG with $\epsilon = 1$ and $\epsilon = 10^{-3}$, which are separated by "/".

According to Table 1 and Table 2, the grouping results of XDG with $\epsilon = 1$ are the same with $\epsilon = 10^{-1}$ on the 20 benchmark functions except f_{11} and f_{16}. On f_{11} and f_{16}, XDG identifies all of the decision variables as separable variables. The reason may be that f_{11} and f_{16} are very smooth functions. The fitness difference between two candidate solutions in the search space is less than 1. Therefore, $\epsilon = 1$ is too large to identify interacting variables.

When $\epsilon = 10^{-3}$, the grouping accuracy of XDG on 18 out of 20 benchmark functions is 100%. On function f_{13}, two non-separable groups are formed with only 50 interacting decision variables correctly identified. The reason for this may be that the computational errors in the system is large compared with the value of ϵ: 10^{-3}. XDG unexpectedly assigns some separable decision variables into the non-separable group. Besides, the 10 groups of non-separable decision variables are assigned to the same group. Note that the second stage of XDG is to learn indirect interaction. An unexpected grouping of interacting decision variables in the first stage may result in two non-separable groups merged as one. It will cause a rapid deterioration of the grouping accuracy. Similar results can be found with f_{18}, where XDG places all of the 20 groups of interacting decision variables

into the same group. Therefore, the grouping accuracy of XDG on f_{18} is 5%.

In sum, the selection of ϵ should not be too large or too small. A large value of ϵ will fail to identify some of the interacting decision variables. A small value of ϵ will result in fault grouping of interacting decision variables. The value 10^{-1} is used for XDG in the rest of the paper.

4.3 DECC Comparison

This section analyses the experimental results of the DECC algorithm with our proposed extended differential grouping method, DECC-XDG, on the 20 benchmark functions. The performance of DECC-XDG is compared with DECC-DG and DECC-D.

On fully separable functions, DECC-D outperforms DECC-DG and DECC-XDG significantly. The reason may be that DG and XDG put all of the separable decision variables into the same sub-component. When the number of separable variables is large (hundreds and thousands), it may not be a good choice to place all of the separable variables into the same sub-component [21, 19]. An alternative decomposition approach could be to treat each separable variable as one sub-component, however, such an approach is not ideal. The intermediate decomposition between these two extreme cases has been shown to be more efficient [15]. Another reason may be that the number of function evaluations used by DECC-XDG or DECC-DG in the decomposition phase is too large, resulting in few function evaluations left for the evolutionary optimization phase. According to Table 1, the number of function evaluations used by DECC-XDG/DG in the decomposition phase is 1001000, which is approximately equal to 1/3 of the maximal number of function evaluations (3×10^6). Note that the performances of DECC-XDG and DECC-DG on the 3 separable functions are almost the same.

On f_8, f_{13}, f_{18} and f_{20}, DECC-XDG outperforms DECC-DG significantly. Note that all the of four functions have indirect interaction between decision variables. XDG achieves perfect decomposition on the four benchmark functions, while the decomposition accuracy of DG is low according to Table 1. Although the number of function evaluations used by DECC-XDG in the decomposition phase is greater than that of DECC-DG (See Table 1), DECC-XDG still significantly outperforms DECC-DG on the four benchmark functions. For f_{18}, DECC-XDG used 25 times the number of function evaluations used by DECC-DG in the decomposition phase. However, the mean of the best solutions found by DECC-XDG is 2.60e+03, while the mean of the best solutions found by DECC-DG is 1.44e+10. The former is much better than the latter, which demonstrates the effectiveness of the extension in XDG. It also shows that an ideal decomposition can significantly improve the performance of a CC algorithm.

On functions without indirect interactions such as f_{15}, f_{16}, f_{17}, DECC-XDG achieves the same or slightly worse results than DECC-DG. Note that the decomposition accuracy achieved by DECC-XDG and DECC-DG are both the same on these functions. However the number of function evaluations used by DECC-XDG in the decomposition phase is much greater than that of DECC-DG. Therefore, the number of function evaluations left for DECC-XDG to optimize each sub-component is much less than that of DECC-DG.

On function f_{20}, DECC-XDG successfully places all of the 1000 interacting decision variables into the same sub-component. However, the quality of the best solution found by DECC-XDG is worse than that of DECC-D. It may be an indication that assigning all of the interacting decision variables into the same sub-component may not be a good decomposition method when the number of interacting variables is large.

Table 3: The comparison between the performances of DECC-XDG, DECC-DG and DECC-D on the 20 benchmark functions. The significant better performances are highlighted in bold (Wilcoxon test with $\alpha = 0.05$)

Func	Num	DECC-XDG	DECC-DG	DECC-D
f_1	Mean	2.23e+04	1.12e+04	**4.07e-24**
	Std	8.01e+04	3.37e+04	1.75e-23
f_2	Mean	4.44e+03	4.42e+03	**2.82e+02**
	Std	1.64e+02	1.59e+02	2.40e+01
f_3	Mean	1.66e+01	1.67e+01	**1.52e-13**
	Std	4.12e-01	3.05e-01	8.48e-15
f_4	Mean	**7.84e+11**	4.63e+12	4.12e+12
	Std	1.67e+11	1.35e+12	1.46e+12
f_5	Mean	**1.68e+08**	1.98e+08	2.48e+08
	Std	1.74e+07	4.58e+07	4.79e+07
f_6	Mean	**1.63e+01**	**1.62e+01**	5.34e+07
	Std	3.28e-01	2.82e-01	8.79e+07
f_7	Mean	**1.39e+03**	1.63e+04	6.89e+07
	Std	2.61e+03	8.93e+03	4.96e+07
f_8	Mean	**4.78e+05**	2.51e+07	1.09e+08
	Std	1.32e+06	2.54e+07	4.87e+07
f_9	Mean	1.12e+08	**5.60e+07**	6.13e+07
	Std	1.13e+07	6.59e+06	6.28e+06
f_{10}	Mean	**5.31e+03**	**5.22e+03**	1.29e+04
	Std	1.55e+02	1.28e+02	2.27e+02
f_{11}	Mean	1.04e+01	9.94e+00	**1.55e-13**
	Std	1.15e+00	9.57e-01	8.19e-15
f_{12}	Mean	1.24e+04	**2.83e+03**	4.30e+06
	Std	2.32e+03	9.92e+02	1.79e+05
f_{13}	Mean	**1.21e+03**	5.35e+06	**1.19e+03**
	Std	2.25e+02	4.89e+06	5.02e+02
f_{14}	Mean	5.83e+08	3.43e+08	**1.93e+08**
	Std	4.11e+07	2.23e+07	1.06e+07
f_{15}	Mean	**5.91e+03**	**5.84e+03**	1.60e+04
	Std	7.56e+01	8.93e+01	4.24e+02
f_{16}	Mean	1.81e-08	**7.32e-13**	1.70e+01
	Std	1.57e-09	4.62e-14	8.48e+01
f_{17}	Mean	1.26e+05	**3.99e+04**	7.48e+06
	Std	7.47e+03	1.80e+03	4.03e+05
f_{18}	Mean	**1.41e+03**	1.44e+10	3.32e+03
	Std	1.88e+02	2.50e+09	7.09e+02
f_{19}	Mean	**1.59e+06**	1.72e+06	2.32e+07
	Std	4.96e+04	6.83e+06	5.56e+06
f_{20}	Mean	5.55e+05	6.69e+10	**1.18e+03**
	Std	1.75e+06	9.24e+09	8.25e+01

5. CONCLUSION

In this paper, we have investigated the effects of decision variable interaction and decomposition methods within the cooperative co-evolution framework. We have described two alternative types of variable interactions – direct interaction and indirect interaction – that may appear in the decision

variable space of large scale optimization problems. Importantly, we have shown that the state-of-art decomposition method DG can not capture indirect interaction, and this in turn can be used to explain relatively poor performance on some of the benchmark functions. As a result, we have introduced an extended differential grouping method, XDG, to address this limitation. Results from comprehensive numerical simulation experiments clearly illustrated that XDG can achieve perfect decomposition on all of the benchmark functions investigated. When XDG was embedded within a cooperative co-evolutionary framework, it achieves significantly better performance than DECC-DG and DECC-D on the benchmark functions with indirect interaction. It also achieves comparable performance with DECC-DG on the benchmark functions without indirect interaction.

6. ACKNOWLEDGMENT

The authors would like to thank Xiaodong Li for providing the source code of the differential grouping method and the DECC algorithm on the website.

7. REFERENCES

[1] W. Chen, T. Weise, Z. Yang, and K. Tang. Large-scale global optimization using cooperative coevolution with variable interaction learning. In *Lecture Notes in Computer Science*, pages 300–309, 2010.

[2] Y. Davidor. Epistasis variance: Suitability of a representation to genetic algorithms. *Complex Systems*, 4:369–383, 1990.

[3] D. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex systems*, 3:493–530, 1989.

[4] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*, volume Addison-We. 1989.

[5] D. E. Goldberg, K. Deb, H. Kargupta, and G. Harik. Rapid Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms,(Urbana, USA)*, pages 56–64, 1993.

[6] N. Hansen. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary computation*, 11(1):1–18, 2003.

[7] H. Kargupta. The performance of the gene expression messy genetic algorithm on real test functions. *Proceedings of IEEE International Conference on Evolutionary Computation*, 1996.

[8] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi. Scaling up fast evolutionary programming with cooperative coevolution. *Proceedings of the 2001 Congress on Evolutionary Computation*, 2, 2001.

[9] H. Mühlenbein, J. Bendisch, and H. M. Voigt. From recombination of genes to the estimation of distributions II. Continuous parameters. *Parallel Problem Solving from Nature - PPSN IV. Springer Berlin Heidelberg*, pages 188–197, 1996.

[10] M. Munetomo and D. Goldberg. A genetic algorithm using linkage identification by nonlinearity check. *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics*, 1, 1999.

[11] M. Munetomo and D. E. Goldberg. Linkage identification by non-monotonicity detection for overlapping functions. *Evolutionary computation*, 7:377–398, 1999.

[12] M. Omidvar, X. Li, Y. Mei, and X. Yao. Cooperative co-evolution with differential grouping for large scale optimization. *IEEE Transcations on Evolutionary Computation*, 18(3):378–393, 2014.

[13] M. N. Omidvar, X. Li, Z. Yang, and X. Yao. Cooperative co-evolution for large scale optimization through more frequent random grouping. In *IEEE Congress on Evolutionary Computation*, 2010.

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

[15] M. N. Omidvar, Y. Mei, and X. Li. Optimal Decomposition of Large-Scale Separable Continuous Functions for Cooperative Co-evolutionary Algorithms. *IEEE Congress on Evolutionary Computation*, 2014.

[16] M. Potter and K. D. Jong. A cooperative coevolutionary approach to function optimization. *Parallel Problem Solving from Nature - PPSN III*, pages 249 – 257, 1994.

[17] S. Rochet, G. Venturini, M. Slimane, and E. E. Kharoubi. A critical and empirical study of epistasis measures for predicting GA performances: a summary. *Artificial evolution*, 1998.

[18] D.-I. Seo and B.-R. Moon. An information-theoretic analysis on the interactions of variables in combinatorial optimization problems. *Evolutionary computation*, 15:169–198, 2007.

[19] Y.-j. Shi, H.-f. Teng, and Z.-q. Li. Cooperative Co-evolutionary Differential Evolution for Function Optimization. In *Advances in Natural Computation*, pages 1080–1088. 2005.

[20] K. Tang, X. Yáo, and P. Suganthan. Benchmark functions for the CEC'2010 special session and competition on large scale global optimization. *Rapport technique, USTC, Nature Inspired Computation and Applications Laboratory*, (1):1–23, 2010.

[21] F. VandenBergh and A. Engelbrecht. A Cooperative Approach to Particle Swarm Optimization. *Evolutionary Computation, IEEE Transactions on*, 8:225–239, 2004.

[22] T. Weise, R. Chiong, and K. Tang. Evolutionary Optimization: Pitfalls and Booby Traps. *Journal of Computer Science and Technology*, 27(5):907–936, Nov. 2012.

[23] Z. Yang, K. Tang, and X. Yao. Large scale evolutionary optimization using cooperative coevolution. *Information Sciences*, 178(15):2985–2999, Aug. 2008.

[24] Z. Yang, K. Tang, and X. Yao. Self-adaptive differential evolution with neighborhood search. In *2008 IEEE Congress on Evolutionary Computation, CEC 2008*, pages 1110–1116, 2008.