Enhanced Differential Grouping for Large Scale Optimization

Mohamed A. Meselhi, Ruhul A. Sarker, Daryl L. Essam and Saber M. Elsayed
School of Engineering and Information Technology, University of New South Wales at Canberra, Canberra, 2600, Australia

Keywords: Evolutionary Algorithms, Cooperative Coevolution, Problem Decomposition, Large Scale Global Optimization.

Abstract: The curse of dimensionality is considered a main impediment in improving the optimization of large scale problems. An intuitive method to enhance the scalability of evolutionary algorithms is cooperative co-evolution. This method can be used to solve high dimensionality problems through a divide-and-conquer strategy. Nevertheless, its performance deteriorates if there is any interaction between subproblems. Thus, a method that tries to group interdependent variables in the same group is demanded. In addition, the computational cost of current decomposition methods is relatively expensive. In this paper, we propose an enhanced differential grouping (EDG) method, that can efficiently uncover separable and nonseparable variables in the first stage. Then, nonseparable variables are furthermore examined to detect their direct and indirect interdependencies, and all interdependent variables are grouped in the same subproblem. The efficiency of the EDG method was evaluated using large scale global optimization benchmark functions with up to 1000 variables. The numerical experimental results indicate that the EDG method efficiently decomposes benchmark functions with fewer fitness evaluations, in comparison with state-of-the-art methods. Moreover, EDG was integrated with cooperative co-evolution, which shows the efficiency of this method over other decomposition methods.

1 INTRODUCTION

In real-world problems (such as computational chemistry, design problems, operations research and biological applications), the growing number of decision variables is considered the main reason for the increased complexity of solving large scale global optimization (LSGO) problems. Firstly, the computational cost of solving these problems using traditional evolutionary algorithms (EAs) is often excessively expensive (Dong et al., 2013). In addition, the performance of these problems deteriorates because of expansion in the search space, which increases exponentially with increases in problem dimension. Furthermore, the complexity of such problems usually leads to a local optimum (Bhattacharya et al., 2016).

There has been great interest in handling LSGO, thus several approaches have been employed in order to solve problems with a large number of decision variables, including but not limited to cooperative co-evolution (CC) (Potter and De Jong, 1994), memetic algorithms (Molina et al., 2011), initialization (Kazimipour et al., 2014; Segredo et al., 2018) and parallelization(Meselhi et al., 2017). The decomposition approach is considered as the first attempt for addressing the curse of dimensionality, which is based on a divide-and-conquer strategy.

The classic CC approach decomposes the LSGO problems into smaller dimension subproblems, and EAs are used to optimize each subproblem cooperatively. This approach has been used successfully to solve different LSGO problems, including combinatorial (Mei et al., 2014), continuous (Omidvar et al., 2014), constrained (Sayed et al., 2015) and multi-objective (Cao et al., 2017). However, the major drawback in the CC approach is that its performance potentially decreases when solving nonseparable optimization problems, due to interdependencies among their subproblems (Salomon, 1996). In nonseparable problems, it was shown that the changing of one subcomponent will lead to a deformation in other interdependent subcomponents' fitness landscapes (Kauffman and Johnsen, 1991). Thus, decomposition techniques that are capable of identifying interacting variables, and so group them into independent subproblems, are desired to improve the performance of the CC approach with large scale optimization problems.

Basically, decomposition methods that have been proposed to use variable grouping for LSGO problems, can be classified into two general approaches,

Meselhi, M., Sarker, R., Essam, D. and Elsayed, S.
Enhanced Differential Grouping for Large Scale Optimization.
DOI: 10.5220/0006938902170224
In *Proceedings of the 10th International Joint Conference on Computational Intelligence (IJCCI 2018)*, pages 217-224
ISBN: 978-989-758-327-8

i.e. static and dynamic grouping methods. In the static grouping method, an n dimension LSGO problem is decomposed into s subproblems of size k in the beginning of CC, and the arrangement of variables in each subproblem remains the same throughout the optimization stage. This method has shown its efficiency only on small dimension problems (i.e. up to 100).

In the second method, dynamic grouping, each subproblem structure is changed dynamically over the optimization process. Decomposition approaches in this method fall into two main categorizes, namely, fixed and automatic decomposition methods. The fixed decomposition method has a fixed subproblem size, that needs to be specified manually; for instance, random grouping (RG) (Yang et al., 2007) and delta grouping [Omidvar et al., 2010b]. The main drawbacks to these approaches is that the values of s and k do not often align with the structure of variables interactions. However, in the automatic decomposition method, the number of subproblems and their sizes is determined automatically, depending on the interaction between an objective function's variables; for example, Variable Interactive Learning (VIL) (Chen et al., 2010), correlation based Adaptive Variable Partitioning (AVP) (Ray and Yao, 2009) and differential grouping (DG) (Omidvar et al., 2014). However, the computational cost of these approaches in the decomposition stage is relatively expensive. In this paper, we propose an enhanced differential grouping (EDG) method. EDG begins by isolating separable variables from nonseparable variables. This separation saves a significant amount of computational cost that would otherwise be wasted in detecting interdependencies in separable variables. Then, nonseparable variables are further examined to detect any interdependencies among them. The EDG method is evaluated using the CEC'2010 benchmark problems (Tang et al., 2009), and the experimental results show 100% grouping accuracy in a smaller number of fitness evaluations, in comparison with state-of-the-art decomposition methods.

The rest of this paper is organized as follows. Section 2 presents the theoretical basis and introduces related work. The proposed algorithm is described in Section 3. Section 4 presents and discusses the experimental studies. Section 5 concludes the paper.

2 LITERATURE REVIEW

In this section, the theoretical bases of separability and variable interaction are presented. This is followed by an overview of various decomposition methods that have been proposed for CC in the literature.

2.1 Separability and Variable Interaction

Separability indicates the degree of interaction between decision variables. A partially separable function is defined as follows: given $f(X)$, where $X = x_1,...,x_n$. If there exists m subproblems $(k_1,...,k_m)$ so that:

$$f(x) = \sum_{i=1}^{m} f_i(x_{k_i});\ 2 \leq m \leq n, \tag{1}$$

then $f(x)$ is a partially separable function.

If $m = n$ (i.e., $k_1,...,k_m$ are one-dimensional subproblems), then $f(x)$ is a fully separable function.

The interaction between two decision variables can be either direct or indirect. Consider the following example:

$$f(X) = x_1 \cdot x_3 + x_2 \cdot x_3 \tag{2}$$

both $x_1 \leftrightarrow x_3$ and $x_2 \leftrightarrow x_3$ have direct interaction with each other, denoted by $\leftrightarrow$. However, there is no direct interaction between x_1 and x_2; instead both of them interact indirectly through x_3. Definitions of direct and indirect interactions are illustrated as follows (Sun et al., 2015). Let $f(x)$ be an additively separable function , and $X^*$is a candidate solution in the decision space. If

$$f'_{x_i x_j}(X^*) \neq 0, \tag{3}$$

then the decision variable x_i and x_j have a direct interaction between them. If

$$f'_{x_i x_j}(X^*) = 0, \tag{4}$$

and there is an indirect link between x_i and x_j through other decision variables in the decision space, then the decision variables x_i and x_j have an indirect interaction between them. Based on the definition of interaction in Equation (3), and if there is no indirect interaction with other variables, then x_i and x_j are independent.

2.2 Cooperative Co-evolution (CC)

Cooperative Co-evolution (CC) is the first attempt to solve LSGO problems using a divide-and-conquer strategy. It decomposes a high-dimensional problem into several smaller dimension subproblems. The typical CC framework has three stages: decomposition, optimization and cooperation. In the decomposition stage, the high-dimension problem is decomposed into several smaller dimension subproblems. For example, the initial CC methods are the one-dimension based- (Potter and De Jong, 1994) and splitting-in-half strategies (Potter and Jong, 2000). In the former method, an n dimension problem is divided

into n one-dimensional subproblems, while the latter method divides an n dimension problem into two $\frac{n}{2}$ subproblems.

In the optimization stage, each subproblem is optimized independently by a traditional optimization algorithm for a certain number of generations. In the classical CC framework (Potter and De Jong, 1994), subproblems evenly share the available computational resources in a round-robin strategy. It has been shown recently that contribution-based CC (Yang et al., 2017) can allocate the available computational resources effectively, by increasing the assigned resources to the subproblems with a higher contribution to fitness improvement. Finally, the cooperative stage exchanges information among all subproblems, and the final solution is constructed by merging solutions from all subproblems.

2.3 Classification of Decomposition Methods

2.3.1 Static Grouping

Static grouping decomposes LSGO problems in the beginning of CC. The CC framework was integrated with Fast Evolutionary Programming (FEP) to solve LSGO problems with 100-1000 variables called FEPCC (Liu et al., 2001). The obtained results showed the inability of a traditional CC framework to deal with nonseparable functions.

Van den Bergh and Engelbrecht (Van den Bergh and Engelbrecht, 2004) incorporated Particle Swarm Optimization to a CC framework called CPSO. An n dimension problem is decomposed into k subproblems of size s. However, CPSO were tested on problems with up to 30 dimensions. Shi et al. (Shi et al., 2005) presented cooperative co-evolutionary differential evolution (CCDE), which partitioned the search space into two equally-sized subproblems. Thus, this decomposition method does not maintain its performance when dimensionality increases.

The Cooperative Micro-Differential Evolution (COMDE) (Parsopoulos, 2009) algorithm divides an LSGO problem into a set of subproblems of size 5, which are easier to evolve. Nevertheless, increasing the number of subproblems leads to expensive computational cost. The interdependencies among the subproblems are not considered in the static grouping methods, which often affect optimization performance. It is clear that the static grouping method is ineffective on partially separable or nonseparable large scale problems.

2.3.2 Dynamic Grouping

In the dynamic grouping method, instead of using static grouping, the arrangement of variables is dynamically changed to deal with the variable interactions. There are two categories based on the characteristics of subproblems, namely a) fixed and b) automatic decomposition methods.

(a) Fixed decomposition methods

Yang et al. (Yang et al., 2007) proposed the random grouping decomposition strategy. In each generation, the arrangement of variables is randomly grouped into k s-dimensional subproblems. Although this method achieved good performance on a set of LSGO problems with dimension of 500 and 1000 (Yang et al., 2008a), it has been shown that the probability of grouping more than four interacting variables in one subproblem reaches approximately 0 (Omidvar et al., 2010).

Ray and Yao (Ray and Yao, 2009) developed a CC Algorithm using AVP, called CCEA-AVP, that detects/measures variables interaction using the Pearson correlation coefficient. All the decision variables are optimized for 5 generations, and then the correlation coefficients that depend on the top 50% of the current population are calculated. According to the obtained correlation coefficients, the variables are grouped into two subproblems, with correlation coefficients greater than a threshold value in one subproblem and all the rest in another subproblem. This method outperforms the standard CC method (Potter and De Jong, 1994) on solving nonseprable problems. However, it does not detect nonlinear interactions among variables (Sayed et al., 2012).

A different method, called delta grouping [Omidvar et al., 2010b], sorts the decision variables based on their absolute magnitude of change across the population between two consecutive generations. Despite the superiority of this method over the random grouping method on solving CEC'2010 benchmark problems (Tang et al., 2009), it has low performance on solving problems with more than one group of nonseprable variables.

Elsayed et al. (2012) presented a Dependency Identification (DI) technique, which divides a LSGO problem into m smaller subproblems of V dependent variables. The goal of this method is to find the best arrangement of variables that minimizes the difference between the evaluation of the complete solution vector $F(x)$ and the sum of each subproblem $\sum_{k=1}^{m} f_k(x_v)$, $v = [1, V]$. Then, subproblems are optimized using a memetic algorithm. DI outperformed random grouping on 8 out of 12 problems. However, a random arrangement of variables is generated at each

iteration; the information from the current arrangement is ignored.

A major drawback of these techniques is that a predetermined number of subproblems needs to be specified. However, the appropriate size of each subproblem depends on each problem's characteristics. For separable variable problems, small sized sub-problems are easier to optimise, while for non-separable problems, a large size subproblem is useful by increasing the probability of the existence of dependent variables in the same subproblem. Therefore, a decomposition algorithm that is automatically able to identify the best number of subproblems and their sizes, based on a problem's characteristics, is required.

(b) Automatic decomposition methods

To the best of our knowledge, Cooperative Co-evolution with Variable Interaction Learning (CCVIL) (Chen et al., 2010) is the first attempt that aims at constructing subproblems based on a problem' characteristics. CCVIL assumes that all decision variables are independent; thus, it starts by decomposing a problem of size N into N one-dimensional subproblems. Then it discovers pairwise interaction between decision variables by using non-monotonicity detection (Munetomo and Goldberg, 1999) . It then merges them into common groups if they affect each other. The CCVIL method is more efficient than fixed grouping methods in identifying variable interactions. However, this method creates N subproblems and so is computationally expensive. In addition, it uses up to 60% of available fitness evaluations for identifying variable interactions.

Omidvar et al. (Omidvar et al., 2014) presented an automated decomposition algorithm, called Differential Grouping (DG), which is able to recognize variable interactions by monitoring the effect of perturbing decision variables on fitness value. The interaction between the first decision variable and all other decision variables is checked in a pair-wise fashion. If any interaction is identified, the algorithm locates the interacting variables in the same subproblem. Otherwise, the first variable is considered a separable variable. This process is repeated until there are no any decision variables left. DG has achieved superior performance over CCVIL on the CEC'2010 benchmark problems (Tang et al., 2009). However, it has been shown that DG is not able to correctly detect overlapping functions. In addition to this, it is sensitive to a threshold parameter which needs to be predetermined by the user. Moreover, the computational cost for fully separable n-dimensional functions is relatively high $O(n^2)$.

Sun et al. (Sun et al., 2015) proposed an improved version of DG, extended differential grouping (XDG), that can address the problem of detecting overlapping functions. After detecting the direct interacting decision variables with the same DG method, any indirect interactions are determined between overlapped subproblems. Finally, both direct and indirect interacting decision variables are grouped into the same subproblems. Despite its improved performance of identifying interactions on CEC'2010 benchmark functions, it is also usually computationally expensive ($O(n^2)$).

3 ENHANCED DIFFERENTIAL GROUPING

In this section, the proposed decomposition method is described in detail. The proposed algorithm starts by separating separable variables from nonseparable variables. Then, in the second stage, directly interacting decision variables are identified. Finally, indirect interaction between decision variables is detected. Thus, the proposed algorithm has three main stages.

Based on the theoretical definition of variable interaction in Equations (3) and (4), the following procedures can be used to identify interaction between any two groups of decision variables (Sun et al., 2017):

1. Calculate $\Delta_1 = f(x^* + l_1u_1 + l_2u_2) - f(x^* + l_2u_2)$.
2. Calculate $\Delta_2 = f(x^* + l_1u_1) - f(x^*)$.
3. An interaction is detected if the difference between Δ_1 and Δ_2 is greater than a threshold ε.

where u_1 and u_2 are two unit vectors, l_1 and l_2 are two real numbers > 0, and x^* is a candidate solution in the search space.

In the first stage, the algorithm begins by detecting both separable and nonseparable variables. Now if separable variables are found, and excluded from the next detection stages, a number of fitness evaluations can be saved. So to identify variable's separability, the interaction between each decision variable and all the remaining variables is examined. The EDG begins by identifying the separability of the first decision variable x_1. All decision variables are set to a lower bound, denoted by $X_{l,l}$ (line 1 of $DeltaDifference$ function), and then x_1 will be perturbed to the upper bound to form $X_{u,l}$ (line 2). The difference in the fitness values at $X_{l,l}$ and $X_{u,l}$ will be calculated, denoted by Δ_1 (line 3). Then all the variables in $X_{l,l}$ and $X_{u,l}$ will be perturbed to the middle of the decision space, except x_1, to form $X_{l,m}$ and $X_{u,m}$ respectively, and the difference in the fitness values between them is then

Algorithm 1: EDG.

Input: f, ub, lb, ε
Output: $sep, nonsep, nonsep\_Groups, FEs$

1: sep = $nonsep$ = [ ];
2: FEs = 0;
3: dim = [1, 2, 3, ..., 1000];
4: **for** $k = 1 : length(dim)$ **do**
5: $X_1 = k$;
6: $X_2 = dim - \{k\}$;
7: $diff = DeltaDifference(X_1, X_2)$;
8: **if** $diff \leq \varepsilon$ **then**
9: $sep = sep \cup X_1$;
10: **else**
11: $nonsep = nonsep \cup X_1$;
12: **end if**
13: **end for**
14: $nonsep\_Groups = \{\}$;
15: **while** $nonsep$ is not empty **do**
16: X_1 = $nonsep(1)$;
17: **for** $j = 2 : length(nonsep)$ **do**
18: X_2 = $nonsep(j)$;
19: $diff = DeltaDifference(X_1, X_2)$;
20: **if** $diff > \varepsilon$ **then**
21: $X_1 = X_1 \cup X_2$;
22: **end if**
23: **end for**
24: $nonsep = nonsep - X_1$;
25: $X_1 = IndirectInteraction(X_1, nonsep)$;
26: $nonsep = nonsep - X_1$;
27: $nonsep\_Groups = \{nonsep\_Groups, X_1\}$;
28: **end while**

calculated, and is denoted by Δ_2 (lines 4-6). The difference of the delta values is calculated, denoted by $diff$ (line 7). If this difference is less than the threshold ε, then x_1 is classified as a separable variable and is placed in the separable group, and the algorithm in this stage moves to the following decision variable x_2. Otherwise, x_1 is considered to be a nonseparable variable (lines 4-13 of Algorithm 1). This process will be repeated until the separability of all the decision variables are identified. For fully separable functions, the interaction detection between decision variables is stopped at the end of this stage, whereas in the case of partial or fully nonseparable functions, further interdependency identification is required in the next stages.

In the second stage, all the variables that interact directly, will be detected and grouped in common subproblems. A pairwise interaction of each nonseparable decision variable, with all other nonseparable decision variables, is examined using the same technique as in the traditional DG method; however, in this stage we are concerned only with nonseparable

Function: DeltaDifference(X_1,X_2).

1: $X_{l,l} = lb$;
2: $X_{u,l} = ub(X_1)$;
3: $\Delta_1 = f(X_{l,l}) - f(X_{u,l})$;
4: $X_{l,m} = (lb(X_2) + ub(X_2))/2$;
5: $X_{u,m} = (lb(X_2) + ub(X_2))/2$;
6: $\Delta_2 = f(X_{l,m}) - f(X_{u,m})$;
7: $diff = \mid \Delta_1 - \Delta_2 \mid$;

Function: IndirectInteraction(X_1,X_2).

1: **while** $DeltaDifference(X_1, nonsep) > \varepsilon$ **do**
2: **if** length $(nonsep) == 1$ **then**
3: $X_1 = X_1 \cup nonsep$;
4: **else**
5: divide $nonsep$ into two equally sized groups $nonsep_1$ and $nonsep_2$;
6: $X_{1_a} = IndirectInteraction(X_1, nonsep_1)$;
7: $X_{1_b} = IndirectInteraction(X_1, nonsep_2)$;
8: $X_1 = X_{1_a} \cup X_{1_b}$;
9: **end if**
10: **end while**

decision variables, which were detected in the first stage. The algorithm checks the direct interaction between any two decision variables x_i and x_j by measuring the difference between Δ_1 and Δ_2 (line 19). If the difference between Δ_1 and Δ_2 is greater than ε, then x_j interacts directly with x_i, and x_j will be located in the same interdependent subproblem with x_i, denoted as X_1. This process will be continued until all variables that interact directly with the decision variable x_i are detected, and the subproblem is shaped. Then, this subproblem will be excluded from the nonseparable group $nonsep$, as shown in lines 16-24.

In the third stage, as far as indirect interaction variables are concerned, the grouped directly interacted variables in X_1 are further recursively examined together with all other remaining nonseparable decision variables, to find any indirect interaction among them. As shown in $IndirectInteraction$ function, if any interaction appears, all the nonseparable decision variables will be divided into two groups with the same size, and then the interaction between X_1 and each group will be identified separately. This process is executed repeatedly until all the indirect decision variables that interact with x_i are detected and merged in $nonsep\_Groups$ with x_i.
Both stages 2 and 3 are repeated for all the remaining decision variables until all subproblems are formed, and finally the EDG returns all the independent subproblems.

Table 1: Decomposition results on CEC'2010 benchmark functions.

Func	EDG		RDG		XDG		DG	
	FEs	accuracy	FEs	accuracy	FEs	accuracy	FEs	accuracy
f_1	**3.01E+03**	**100%**	**3.01E+03**	**100%**	1.00E+06	100%	1.00E+06	100%
f_2	**3.01E+03**	**100%**	**3.01E+03**	**100%**	1.00E+06	100%	1.00E+06	100%
f_3	**5.00E+03**	**100%**	6.00E+03	100%	1.00E+06	100%	1.00E+06	100%
f_4	**3.10E+03**	**100%**	4.21E+03	100%	8.05E+04	-	1.45E+04	100%
f_5	**3.10E+03**	**100%**	4.15E+03	100%	9.98E+05	100%	9.05E+05	100%
f_6	**3.10E+03**	**100%**	5.03E+04	100%	9.98E+05	100%	9.06E+05	100%
f_7	**3.10E+03**	**100%**	4.23E+03	100%	9.98E+05	100%	6.77E+04	68%
f_8	**3.84E+03**	**100%**	5.61E+03	100%	1.21E+05	-	2.32E+04	100%
f_9	**8.48E+03**	**100%**	1.40E+04	100%	9.77E+05	100%	2.70E+05	100%
f_{10}	**8.48E+03**	**100%**	1.40E+04	100%	9.77E+05	100%	2.72E+05	100%
f_{11}	1.50E+04	100%	**1.37E+04**	**100%**	9.78E+05	100%	2.70E+05	99.8%
f_{12}	**8.48E+03**	**100%**	1.43E+04	100%	9.77E+05	100%	2.71E+05	100%
f_{13}	**2.60E+04**	**100%**	2.92E+04	100%	1.00E+06	100%	5.03E+04	31.8%
f_{14}	2.40E+04	100%	**2.06E+04**	**100%**	9.53E+05	100%	2.10E+04	100%
f_{15}	2.40E+04	100%	**2.05E+04**	**100%**	9.53E+05	100%	2.10E+04	100%
f_{16}	2.40E+04	100%	**2.09E+04**	**100%**	9.56E+05	100%	2.11E+04	99.6%
f_{17}	2.40E+04	100%	**2.08E+04**	**100%**	9.53E+05	100%	2.10E+04	100%
f_{18}	6.48E+04	100%	**4.99E+04**	**100%**	9.99E+05	100%	3.96E+04	23%
f_{19}	5.00E+03	100%	6.00E+03	100%	3.99E+03	100%	**2.00E+03**	**100%**
f_{20}	**9.99E+03**	**100%**	5.09E+04	100%	1.00E+06	100%	1.55E+05	28.7%

4 EXPERIMENTS AND RESULTS

To evaluate the performance of the proposed EDG algorithm, the CEC'2010 benchmark problems (Tang et al., 2009) on large-scale global optimization were used. The CEC'2010 benchmark functions includes 20 functions which are grouped into five categories as follows:

1. Fully separable functions (f_1-f_3)
2. Single-group m-nonseparable functions (f_4-f_8)
3. $\frac{D}{2m}$-group m-nonseparable functions (f_9-f_{13})
4. $\frac{D}{m}$-group m-nonseparable functions(f_{14}-f_{18})
5. Fully non-separable functions (f_{19}-f_{20})

where D is the problem's dimension and m is the number of variables in each nonseparable subproblem. In this paper, D and m are set to 1000 and 50, respectively for all benchmark functions.

In the grouping stage, rather than a fixed threshold ε value that does not suit all test functions (Omidvar et al., 2014), an adaptive ε is adopted. The best ε value for each specific function is determined using the magnitude of the objective values in the decision space (Mei et al., 2016):

$$\varepsilon = \alpha \cdot min\{|\, f(x_1)\,|, ..., |\, f(x_K)\,|\},$$

where α is the control coefficient (set to 10^{-10}), and k random solutions $x_1, ..., x_K$ in the decision space (Mei et al., 2016). While for the optimization stage, a variant of Differential Evolution – SaNSDE (Yang et al., 2008b) is used to optimize each subproblem cooperatively. The experimental results are based on 25 independent runs, where the population size is 50, and the maximum number of fitness evaluations, divided between grouping and optimizing stages, is 3×10^6.

4.1 Decomposition Results

The decomposition results of EDG, RDG (Sun et al., 2017), XDG (Sun et al., 2015) and DG (Omidvar et al., 2014) are presented in Table 1, which shows that the EDG and RDG methods achieve 100% grouping accuracy on all the 20 benchmark functions, as they use the same estimation of the threshold ε. In contrast, XDG and DG correctly decompose 18 and 12 benchmark functions, respectively. DG has poor decomposition accuracy on indirect interaction functions.

Table 1 also illustrates the number of fitness evaluations used by each method. For the first three functions that are fully separable functions (f_1-f_3), the number of fitness evaluations used by XDG and DG are 1.00E+06. While both EDG and RDG identify the separability of all decision variables on f_1 and f_2, using 3.01E+03 FEs. EDG uses 5.00E+03 FEs to identify f_3, in comparison to 6.00E+03 FEs for RDG.

For partially separable functions with one nonseparable group of 50 variables (f_4-f_8), EDG uses the smallest number of fitness evaluations to correctly identify the 50 nonseparable variables and 950 separable variables.

Category 2 and 3 contain one and 10 nonseparable group, respectively, each with 50 variables (f_4-f_{13}). EDG again uses the smallest number of fitness evaluations to correctly identify all the 50 nonseparable

variables groups and separable variables group. This holds for all test functions, except for f_{11}, where EDG uses slightly more fitness evaluation than RDG.

For category 4 (partially separable functions that contain 20 nonseparable groups), EDG uses more fitness evaluations than RDG. This is because of the fitness evaluations used in detecting the separable part of the decision variables, that does not exist in this category. In category 5, the DG and EDG methods use the smallest number of fitness evaluations on f_{19} and f_{20}, respectively.

As the aforementioned results show, the EDG method correctly identifies all decision variables on all benchmark functions, while using the smallest number of fitness evaluations on 13 out of 20 benchmark problems.

4.2 Optimization Results

This section shows the performance of EDG, DI, DG and D (delta grouping) methods, when integrated in the DECC cooperative co-evolutionary framework. Table 2 reports the experimental results of the compared decomposition algorithms for 25 independent runs on the CEC'2010 benchmark problems. The EDG and DG methods group all separable variables in one subproblem. Thus, on fully separable functions (f_1-f_3), the D method outperforms both the EDG and DG methods (where subproblem size = 1000).

The EDG method achieved the best mean results on 11 out of 20 optimization functions, namely f_4-f_8, f_9, f_{10}, f_{14}, f_{15}, f_{17}, f_{18}.

This demonstrates that good decomposition can effectively enhance the performance of the optimization stage in the CC framework. On f_{19} and f_{20}, despite the correct decomposition, EDG has worse optimization performance than DI. This is because of its grouping of all 1000 nonseparable decision variables into one interdependent subproblem.

Table 2: Optimization results on CEC'2010 benchmark functions.

Func	Stats	EDG	DI	DG	D
f_1	Mean	2.64E+05	8.28E-06	1.12E+04	**4.07E-24**
	Std	3.19E+05	3.24E-05	3.37E+04	1.75E-23
f_2	Mean	4.14E+03	5.44E+02	4.42E+03	**2.82E+02**
	Std	4.65E+02	1.16E+02	1.59E+02	2.40E+01
f_3	Mean	1.10E+01	6.33E+00	1.67E+01	**1.52E-13**
	Std	6.45E-01	9.38E-01	3.05E-01	8.48E-15
f_4	Mean	**3.10E+10**	2.83E+12	4.63E+12	4.12E+12
	Std	1.42E+10	1.01E+12	1.35E+12	1.46E+12
f_5	Mean	**7.12E+07**	2.44E+08	1.98E+08	2.48E+08
	Std	1.47E+07	3.20E+07	4.58E+07	4.79E+07
f_6	Mean	**1.60E+01**	2.21E+06	1.62E+01	5.34E+07
	Std	7.46E+03	2.88E+05	2.82E-01	8.79E+07
f_7	Mean	**1.31E+04**	7.27E+07	1.63E+04	6.89E+07
	Std	7.46E+03	3.62E+08	8.93E+03	4.96E+07
f_8	Mean	**3.27E+05**	2.15E+07	2.51E+07	1.09E+08
	Std	1.10E+06	2.76E+07	2.54E+07	4.87E+07
f_9	Mean	**2.34E+07**	8.19E+07	5.60E+07	6.13E+07
	Std	8.28E+06	8.59E+06	6.59E+06	6.28E+06
f_{10}	Mean	**3.01E+03**	9.96E+03	5.22E+03	1.29E+04
	Std	2.22E+02	2.54E+03	1.28E+02	2.27E+02
f_{11}	Mean	2.58E+01	9.13E+01	9.94E+00	**1.55E-13**
	Std	2.64E+00	1.75E+01	9.57E-01	8.19E-15
f_{12}	Mean	1.89E+04	2.95E+04	**2.83E+03**	4.30E+06
	Std	9.14E+03	8.59E+06	9.92E+02	1.79E+05
f_{13}	Mean	1.11E+04	2.48E+03	5.35E+06	**1.19E+03**
	Std	4.36E+03	2.54E+03	4.89E+06	5.02E+02
f_{14}	Mean	**2.15E+07**	2.50E+08	3.43E+08	1.93E+08
	Std	2.28E+06	1.75E+01	2.23E+07	1.06E+07
f_{15}	Mean	**2.94E+03**	1.01E+04	5.84E+03	1.60E+04
	Std	2.67E+02	4.44E+03	8.93E+01	4.24E+02
f_{16}	Mean	1.91E+01	2.47E+02	**7.32E-13**	1.70E+01
	Std	2.80E+00	2.47E+01	4.62E-14	8.48E+01
f_{17}	Mean	**7.50E+00**	1.12E+05	3.99E+04	7.48E+06
	Std	1.40E+00	2.46E+04	1.80E+03	4.03E+05
f_{18}	Mean	**1.20E+03**	7.30E+03	1.44E+10	3.32E+03
	Std	1.44E+02	2.93E+03	2.50E+09	7.09E+02
f_{19}	Mean	9.24E+05	**6.16E+05**	1.72E+06	2.32E+07
	Std	8.95E+04	5.17E+04	6.83E+06	5.56E+06
f_{20}	Mean	8.48E+07	2.92E+03	6.69E+10	**1.18E+03**
	Std	2.49E+08	1.83E+02	9.24E+09	8.25E+01

5 CONCLUSIONS

This paper introduced an enhanced differential grouping (EDG) for LSGO problems. In the proposed method, separable and nonseparable variables can be classified in the first stage. Thus, a significant number of fitness evaluations which would be used to identify interdependency among decision variables can be saved. Then, direct and indirect interdependencies of nonseparable variables are detected. Results from numerical experiments indicate that EDG can achieve 100% grouping accuracy on all benchmark functions with fewer fitness evaluations. EDG was also embedded in a DECC framework, where it achieved better performance than state-of-the-art algorithms.

REFERENCES

Bhattacharya, M., Islam, R., and Abawajy, J. (2016). Evolutionary optimization: a big data perspective. *Journal of Network and Computer Applications*, 59:416–426.

Cao, B., Zhao, J., Lv, Z., and Liu, X. (2017). A distributed parallel cooperative coevolutionary multiobjective evolutionary algorithm for large-scale optimization. *IEEE Transactions on Industrial Informatics*, 13(4):2030–2038.

Chen, W., Weise, T., Yang, Z., and Tang, K. (2010). Large-scale global optimization using cooperative coevolution with variable interaction learning. In *International Conference on Parallel Problem Solving from Nature*, pages 300–309. Springer.

Dong, W., Chen, T., Tino, P., and Yao, X. (2013). Scaling up estimation of distribution algorithms for continu-

ous optimization. *IEEE Transactions on Evolutionary Computation*, 17(6):797–822.

Kauffman, S. A. and Johnsen, S. (1991). Coevolution to the edge of chaos: coupled fitness landscapes, poised states, and coevolutionary avalanches. *Journal of theoretical biology*, 149(4):467–505.

Kazimipour, B., Li, X., and Qin, A. K. (2014). A review of population initialization techniques for evolutionary algorithms. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 2585–2592. IEEE.

Liu, Y., Yao, X., Zhao, Q., and Higuchi, T. (2001). Scaling up fast evolutionary programming with cooperative coevolution. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 2, pages 1101–1108. Ieee.

Mei, Y., Li, X., and Yao, X. (2014). Cooperative coevolution with route distance grouping for large-scale capacitated arc routing problems. *IEEE Transactions on Evolutionary Computation*, 18(3):435–449.

Mei, Y., Omidvar, M. N., Li, X., and Yao, X. (2016). A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. *ACM Transactions on Mathematical Software (TOMS)*, 42(2):13.

Meselhi, M. A., Elsayed, S. M., Essam, D. L., and Sarker, R. A. (2017). Fast differential evolution for big optimization. In *Software, Knowledge, Information Management and Applications (SKIMA), 2017 11th International Conference on*, pages 1–6. IEEE.

Molina, D., Lozano, M., Sánchez, A. M., and Herrera, F. (2011). Memetic algorithms based on local search chains for large scale continuous optimisation problems: Ma-ssw-chains. *Soft Computing*, 15(11):2201–2220.

Munetomo, M. and Goldberg, D. E. (1999). Linkage identification by non-monotonicity detection for overlapping functions. *Evolutionary computation*, 7(4):377–398.

Omidvar, M. N., Li, X., Mei, Y., and Yao, X. (2014). Cooperative co-evolution with differential grouping for large scale optimization. *IEEE Transactions on evolutionary computation*, 18(3):378–393.

Omidvar, M. N., Li, X., Yang, Z., and Yao, X. (2010). Cooperative co-evolution for large scale optimization through more frequent random grouping. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE.

Parsopoulos, K. E. (2009). Cooperative micro-differential evolution for high-dimensional problems. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 531–538. ACM.

Potter, M. A. and De Jong, K. A. (1994). A cooperative coevolutionary approach to function optimization. In *International Conference on Parallel Problem Solving from Nature*, pages 249–257. Springer.

Potter, M. A. and Jong, K. A. D. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary computation*, 8(1):1–29.

Ray, T. and Yao, X. (2009). A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 983–989. IEEE.

Salomon, R. (1996). Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms. *BioSystems*, 39(3):263–278.

Sayed, E., Essam, D., and Sarker, R. (2012). Using hybrid dependency identification with a memetic algorithm for large scale optimization problems. In *Asia-Pacific Conference on Simulated Evolution and Learning*, pages 168–177. Springer.

Sayed, E., Essam, D., Sarker, R., and Elsayed, S. (2015). Decomposition-based evolutionary algorithm for large scale constrained problems. *Information Sciences*, 316:457–486.

Segredo, E., Paechter, B., Segura, C., and González-Vila, C. I. (2018). On the comparison of initialisation strategies in differential evolution for large scale optimisation. *Optimization Letters*, 12(1):221–234.

Shi, Y.-j., Teng, H.-f., and Li, Z.-q. (2005). Cooperative co-evolutionary differential evolution for function optimization. In *International Conference on Natural Computation*, pages 1080–1088. Springer.

Sun, Y., Kirley, M., and Halgamuge, S. K. (2015). Extended differential grouping for large scale global optimization with direct and indirect variable interactions. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 313–320. ACM.

Sun, Y., Kirley, M., and Halgamuge, S. K. (2017). A recursive decomposition method for large scale continuous optimization. *IEEE Transactions on Evolutionary Computation*.

Tang, K., Li, X., Suganthan, P., Yang, Z., and Weise, T. (2009). Benchmark functions for the cec 2010 special session and competition on large-scale global optimization: Nature inspired computation and applications laboratory, university of science and technology of china. *Applicat. Lab., Univ. Sci. Technol. China, Hefei, China, Tech. Rep*.

Van den Bergh, F. and Engelbrecht, A. P. (2004). A cooperative approach to particle swarm optimization. *IEEE transactions on evolutionary computation*, 8(3):225–239.

Yang, M., Omidvar, M. N., Li, C., Li, X., Cai, Z., Kazimipour, B., and Yao, X. (2017). Efficient resource allocation in cooperative co-evolution for large-scale global optimization. *IEEE Transactions on Evolutionary Computation*, 21(4):493–505.

Yang, Z., Tang, K., and Yao, X. (2007). Differential evolution for high-dimensional function optimization. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 3523–3530. IEEE.

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

Yang, Z., Tang, K., and Yao, X. (2008b). Self-adaptive differential evolution with neighborhood search. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 1110–1116. IEEE.