Cooperative Co-evolution with Delta Grouping for Large Scale Non-separable Function Optimization

Mohammad Nabi Omidvar, *Member, IEEE* and Xiaodong Li, *Senior Member, IEEE* and Xin Yao, *Fellow, IEEE*

***Abstract*— Many evolutionary algorithms have been proposed for large scale optimization. Parameter interaction in non-separable problems is a major source of performance loss specially on large scale problems. Cooperative Co-evolution(CC) has been proposed as a natural solution for large scale optimization problems, but lack of a systematic way of decomposing large scale non-separable problems is a major obstacle for CC frameworks. The aim of this paper is to propose a systematic way of capturing interacting variables for a more effective problem decomposition suitable for cooperative co-evolutionary frameworks. Grouping interacting variables in different subcomponents in a CC framework imposes a limit to the extent interacting variables can be optimized to their optimum values, in other words it limits the improvement interval of interacting variables. This is the central idea of the newly proposed technique which is called *delta method*. *Delta method* measures the averaged difference in a certain variable across the entire population and uses it for identifying interacting variables. The experimental results show that this new technique is more effective than the existing random grouping method.**

I. Introduction

Numerous metaheuristic algorithms have been developed for continuous global function optimization. A major issue with these techniques is their scalability to higher dimensions. It has been shown that the performance of these algorithms drops rapidly as the dimensionality of the problem increases[1]. This is mainly due to exponential growth in the size of search space. Cooperative Coevolution(CC) is a popular technique in Evolutionary Algorithms(EAs) for large scale optimization. Using a CC approach the decision variables of the problem are divided into smaller subcomponents each of which is optimized using a separate EA.

Despite its success in solving many optimization problems, CC loses its efficiency when applied to non-separable problems. Non-separable problems are those subclass of optimization problems where a proportion of the decision variables have interaction amongst themselves. When these interacting variables are placed in different subcomponents there would be a major decline in the overall performance of the algorithms. This problem calls for new techniques which are capable of capturing the interacting variables and grouping them in one subcomponent.

M. N. Omidvar and X. Li are with the Evolutionary Computing and Machine Learning Group(ECML), the School of Computer Science and IT, RMIT University, VIC 3001, Melbourne, Australia (emails: momidvar@cs.rmit.edu.au, xiaodong.li@rmit.edu.au).

X. Yao is with the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), the School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK (e-mail: x.yao@cs.bham.ac.uk).

Recently a new technique called random grouping has been proposed by Yang et al. [2] which shows significant improvement over previous CC techniques for large scale non-separable optimization problems. In random grouping, every decision variable is randomly assigned to any of the subcomponents with equal probability and the whole process is repeated at the beginning of every cycle. It has been shown that using this technique the probability of grouping two interacting variables in the same subcomponent will increase significantly[2]. Unfortunately this probability will drop significantly when there are more than two interacting variables.

Ray and Yao proposed a Cooperative Co-evolutionary Algorithm using Correlation based Adaptive Variable Partitioning technique (CCEA-AVP)[3] in which they optimized all of the decision variables in one subcomponent for 5 generations before calculating the correlation coefficients of the top 50% of individuals in the population. The variables with correlation coefficient value greater than a predetermined threshold are grouped in one subcomponent and the rest in another subcomponent. It has been shown that this technique performs better than traditional CCEAs on many non-separable benchmark functions.

In this paper we propose a new technique for capturing the interacting variables and grouping them in one subcomponent. As it has been shown by Salomon in [4] the improvement interval of a function shrinks significantly when it has high eccentricity and is not aligned with the coordinate axes (in other words, when the variables are interacting with each other). Improvement interval of a variable is the interval in which the fitness value could be improved while all the other variables are kept constant. This is further explained in Section III. In non-separable functions, when two interacting variables are grouped in separate subcomponents, there would be a limit to the extent each variable can be improved towards its optimal value due to the rotation of the function. We speculated that measuring the amount of change in each of the decision variables in every iteration can lead to identification of interacting variables. We called this amount of change *delta value*. As we mentioned earlier the improvement interval will shrink significantly on a non-separable function so we expect smaller delta values for those variables that have interaction amongst themselves. Based on this idea we proposed a new algorithm that sorts the decision variables based on the magnitude of their delta values and groups the variables with smaller delta values in one subcomponent. Experimental observations revealed that

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

this new technique is very effective in capturing interacting variables compared to previous techniques.

The organization of the rest of this paper is as follows. Section II briefly explains the preliminaries. Section III describes the details of the new technique for capturing interacting variables. Section IV demonstrates and analyzes the experimental results. Finally Section V concludes this paper and gives directions on further potential improvements.

II. Preliminaries

A. Cooperative Co-evolution

Cooperative Co-evolution(CC) [5] proposed by Potter and De Jong is a framework for decomposition of problems into smaller subcomponents each of which is evolved using a separate EA. Potter and De Jong incorporated CC into Genetic Algorithm for function optimization and have shown significant improvement over traditional GA. This algorithm is called Cooperative Co-evolutionary Genetic Algorithm(CCGA) [5].

CCGA divides a n-dimensional decision vector into n 1-dimensional subcomponents each of which is optimized using a separate GA in round robin fashion.

In the past decades CC has been incorporated into various evolutionary algorithms such as Evolutionary Programming [6], Evolutionary Strategies[7], Particle Swarm Optimization[8], and Differential Evolution [2], [9].

The divide-and-conquer approach of CC makes it ideal for tackling large scale problems. Liu et al. developed Fast Evolutionary Programming with Cooperative Co-evolution(FEPCC)[6] by which they tackled function optimization problems with up to 1000 dimensions. A major drawback of Potter and De Jong's decomposition strategy is that it doesn't account for variable interactions. Poor performance of FEPCC on non-separable functions supports this idea[6].

Cooperative Particle Swarm Optimization(CPSO)[8] is the first cooperative co-evolutionary PSO which was proposed by van den Bergh and Engelbrecht. CPSO, unlike CCGA decomposes a n-dimensional problem into m s-dimensional subcomponents where s is the number of variables in a subcomponent. CPSO uses a static grouping which means that the arrangement of variables is not changed through the evolutionary process. In order to evaluate the individuals in each of the subcomponents, the variables of every individual are concatenated with the best-fit individuals of other subcomponents to form what is called a *context vector* [8] and then the context vector is passed to fitness function for evaluation.

The first attempt for applying CC to large scale optimization was made by Liu et al. using Fast Evolutionary Programming with Cooperative Co-evolution(FEPCC)[6] where they tackled problems with up to 1000 dimensions, but it converged prematurely for one of the non-separable functions, confirming that Potter and De Jong decomposition strategy is ineffective in dealing with variable interaction.

Shi et al. [9] proposed yet another decomposition strategy in which they divided the decision vector into halves and each half is optimized using Differential Evolution(DE)[1]. Splitting-in-half strategy can barely scale up with the dimensions of the problem. This is because the dimensionality of the two subcomponents will soon go beyond the capabilities of subcomponent optimizers in higher dimensions.

Like CPSO, Yang et al. [2] subdivide a n-dimensional decision vector into m s-dimensional subcomponents, but unlike CPSO they use a dynamic grouping technique called *random grouping*. Random grouping is explained in more details in Section II-B.

B. Random Grouping and Adaptive Weighting

As it was mentioned in Section II Yang et al. [2] proposed random grouping as a simple way of increasing the probability of grouping interacting variables in one subcomponent. Below is an outline of the algorithm proposed in [2] Which is called DECC-G.

1) set $i = 1$ to start a new *cycle*.
2) Randomly split a n-dimensional decision vector into m s-dimensional vectors. This essentially means that any variable has equal chance of being assigned to any of the subcomponents.
3) Optimize the i^{th} subcomponent with a certain EA for a predefined number of Fitness Evaluations(FEs).
4) If $i < m$ then $i++$, and go to Step 3.
5) Construct a weight vector and evolve it using a separate EA for the best, worst, and a random member of the current population.
6) Stop if the maximum number of FEs is reached or go to Step 1 for the next *cycle*

Step 2 of the above algorithm is where the random grouping is invoked. Yang et al. have shown in [2] that the probability of grouping two interacting variables for at least two out of 50 cycles using 10 subcomponents each containing 100 variables is approximately 96.62%. Generally random grouping will increase the probability of assigning two interacting variables into the same subcomponent.

In an attempt to generalize the theorem proposed by Yang et al.[2], Omidvar et al. have shown that the probability of grouping interacting variables into one subcomponent will drop significantly as the number of interacting variables increases[1]. This probability can be calculated using Equation (1).

$$P(X \geq k) = \sum_{r=k}^{N} \binom{N}{r} \left(\frac{1}{m^{v-1}}\right)^{r} \left(1 - \frac{1}{m^{v-1}}\right)^{N-r} \quad (1)$$

where N is the total number of cycles, k is the minimum number of cycles that the interacting variables should be grouped into one subcomponent, m is the number of subcomponents, v is the number of interacting variables that needs to be grouped in one subcomponent and the random variable X is the number of times that v interacting variables are grouped in one subcomponent and since we are interested in the probability of grouping v interacting variables for at least k cycles, X takes the values greater than or equal to k. k is also subject to the following condition $k \leq N$.

Figure 1 plots Equation (1) for $N = 50$, and $N = 10000$ and shows how the probability drops as the number of interacting variables increases. One might think that increasing the frequency of random grouping is the solution to the problem, but this will significantly increase the total number of fitness evaluations. In [1] Omidvar et al. proposed several techniques in saving computational cost and using that for more frequent random grouping. Although the proposed technique significantly improves the performance of DECC-G, the gained performance is insufficient to group more than 7 interacting variables even for only one cycle. This behavior is shown by the dashed line in Figure 1. As it can be seen from the same figure using the techniques described in [1] it is possible to increase the frequency of random grouping from 50 cycles to 10000 cycles using the same number of fitness evaluations, but increasing the frequency of random grouping by a factor of 200 allows for grouping only two more interacting variables into one subcomponent with the same probability compared to the case where only 50 cycles were used. This can be inferred form the two units shift of the probability plot to the right in Figure 1. For better clarity a horizontal line could be imagined at a given probability. For example a horizontal line at the probability of 0.4 will cross the solid line($N = 50$) at approximately $v = 3$, and the dashed line($N = 10000$) at approximately $v = 5$. So increasing the number of cycles from 50 to 10000 allows for grouping only two more interacting variables at any given probability. The focus of this paper is to propose a more effective technique in grouping interacting variables in one subcomponent. Further details about the new technique is provided in Section III.

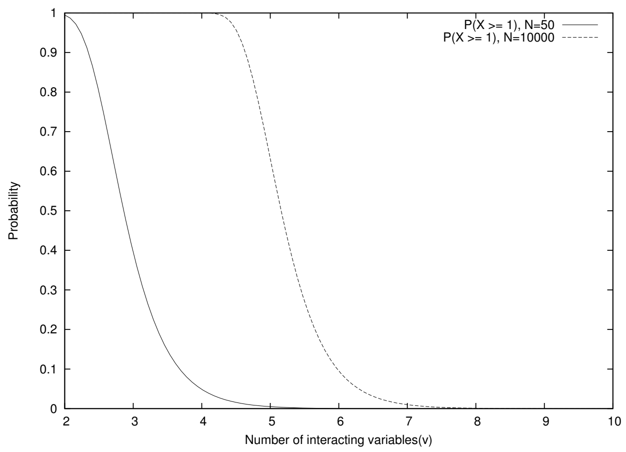


Fig. 1. Increasing the number of interacting variables(v) will significantly decrease the probability of grouping them in one subcomponent, given $n = 1000$ and $m = 10$ [1].

Another subcomponent of DECC-G is *adaptive weighting* as described in Step 5 of the outline of the algorithm earlier in this section. In adaptive weighting a weight vector is constructed using a series of coefficients which are multiplied by the variables of subcomponents before evaluation. The weight vector itself is evolved using a separate EA at the end of each cycle. Adaptive weighting is done in order to co-adapt the interdependent subcomponents. In [1] Omidvar et al. have shown that adaptive weighting is not very effective and in most cases fails to further improve the solution.

C. Self-adaptation of subcomponent sizes

Multilevel Cooperative Co-evolution(MLCC)[10] is an extension of DECC-G which self adapts the subcomponent sizes using their historical performance through the course of evolution. MLCC maintains a set of predetermined subcomponent sizes which are chosen at the start of each cycle based on their performance records which are calculated using special formulas described in [10]. The performance of MLCC is shown to be better than DECC-G in [10].

In this paper we use a simpler self-adaptation technique for subcomponent sizes which was used in [1]. This new technique uses a similar set of subcomponent sizes used in MLCC but the mechanism for choosing the decomposers differs. In this technique the fitness of the best solution is monitored and once there is no further improvement in the fitness value a different decomposer is randomly chosen from the set using a uniform random number generator. This new DECC variant is called DECC-ML which is shown to have a better performance than MLCC in [1].

III. Proposed Techniques

A. Delta Method

As it was described earlier in Section II-B the probability of grouping interacting variables in one subcomponent will decrease rapidly as the number of interacting variables increases (Figure 1). Here we propose a new technique that is capable of grouping interacting variables in a more systematic way for more cycles than what is possible with random grouping.

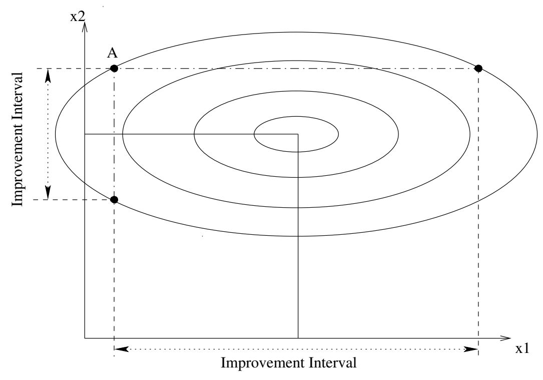
As it was mentioned in Section I CCEA-AVP relies on calculation of correlation matrix for grouping interacting variables. A major disadvantage of CCEA-AVP is that it relies on a splitting-in-half strategy for decomposition of decision variables which has limited scalability to higher dimensions. This is similar to the scalability problem of the algorithm proposed by Shi et al. in [9]. The other issue of CCEA-AVP is that it remains unclear as to what type of correlation coefficient has been used in CCEA-AVP. To the best of our knowledge most statistical correlation coefficients such as Pearson's product-moment coefficient are used to measure the linear dependence between any two variables and it is independent of the slope of the line that they form. In other words two variables might be highly linearly correlated to each other yet they might be completely separable i.e. linear correlation of variables which could be measured using correlation coefficients is not a proper measure for separability of variables in the context of large scale non-separable optimization.

In this paper we propose a new technique to address the shortcomings of random grouping and correlation based methods. The main idea behind this new technique is based on a major property of most of non-separable problems. In cooperative co-evolution the decision vector is decomposed into several smaller subcomponents and in order to optimize and evaluate the individuals within each of the subcomponents, the variables of other subcomponents must be kept fixed. In separable problems each variable can be optimized one at a time and since there is no interaction between the variables it is possible to get as close as possible to the optimal value for that specific variable. Whereas in a non-separable problem, variable interaction, imposes some limitation to the extent a variable can be optimized close to its optimal value. In other words variable interaction reduces the improvement interval of variables specially when they are grouped in different subcomponents. Coordinate rotation is one way of turning a separable problems into a non-separable one[4]. Salomon used this technique to show how the performance of GA can drop significantly when applied to rotated functions specially those with high eccentricity[4].

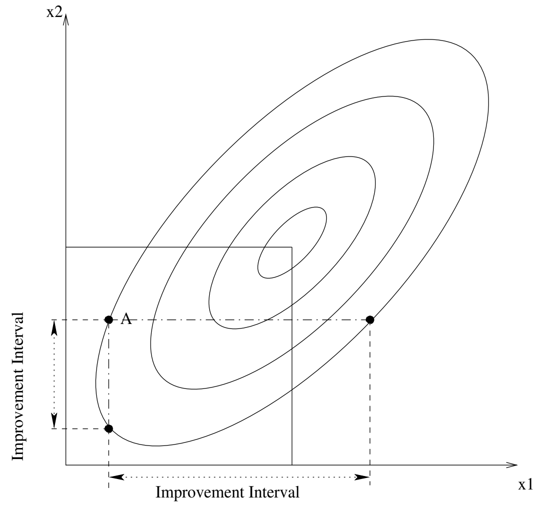
Figure 2 shows how coordinate rotation reduces the improvement interval of interacting variables. Figure 2(a) shows a quadratic function with its principal axis aligned with the coordinate system. As it can be seen the improvement window is relatively large and with some luck it is possible for the CCEA to find the optimal value of each variable before optimizing the other variable. Figure 2(b) is identical to Figure 2(a) except that it is rotated. Point A on the level curve stays in the same location relative to the shape of the function, but as it is depicted in Figure 2(b) coordinate rotation causes the improvement interval of point A to shrink considerably. Moreover, when the interacting variables are grouped in different subcomponents, it becomes increasingly difficult to optimize the variables to their optimal value. Figure 2(b) clearly shows if one of the variables is kept fixed the other variable is confined to reaching only a suboptimal value.

We can use this information of small improvement intervals to identify possible interacting variables in a non-separable problem. When the improvement interval of a variable gets smaller, it creeps towards its optimal value in small successions. So whenever we observe such behavior we can conclude that there might be another variable with a small improvement window. This is not necessarily true all the times, but using this simple heuristic we can significantly increase the probability of grouping interacting variables into one subcomponent. Another advantage of the new technique is its ability to adapt itself to the fitness landscape. It is clear that the degree of non-separability might change over time depending on the area of landscape that is explored by the individuals in the population. This flexibility allows the algorithm to use more suitable decomposition strategy depending on the shape of fitness landscape.

In the algorithm proposed in this paper we calculate the amount of change in each of the dimensions between two consecutive cycles for all of the individuals in the population. We then construct the vector $\Delta = \{\bar{\delta}_1, \bar{\delta}_2, ..., \bar{\delta}_n\}$ where n is the number of dimensions and the elements of the vector are calculated using the following equation:



(a) Quadratic Function



(b) Rotated Quadratic Function

Fig. 2. Coordinate rotation causes the improvement interval to shrink. Figures (b) is the rotated version of function in Figure (a). Point A on the level curves represent the same point before and after rotation. Note that in (b) the global optimum is outside of the improvement window, which makes it much harder for an algorithm to locate the global optimum.

$$\bar{\delta}_i = \frac{\sum_{j=1}^{N_{pop}} \delta_{i,j}}{N_{pop}}, i \in \{1, 2, ..., n\} \quad (2)$$

where N_{pop} is the population size, and $\delta_{i,j}$ denotes the delta value of the j^{th} individual on the i^{th} dimension. This is a rough estimation of improvement interval in every dimensions which used for capturing the interacting variables.

We can then sort the decision variables based on the magnitude of their corresponding delta value in descending order, and finally they are divided into predetermined equally-sized subcomponents. Based on the idea of a small improvement interval when a small delta value is observed there is a high probability that another variable exists with a relatively small delta value that has interaction with the former variable. This algorithm is called Differential Evolution with Cooperative Co-evolution using Delta-Grouping(DECC-D) which is outlined below.

1) set $i = 1$ to start a new *cycle*.
2) Initialize the Δ vector to zero. This means that the delta is not used for arrangement of variables in the first cycle.
3) Divide the decision variables into predefined subcomponent. In the first cycle Δ vector is initialized to zero so the variables will preserve their original order.
4) Optimize the i^{th} subcomponent with a certain EA for one only one iteration. Note that this differs from DECC-G[2] where the subcomponent optimizers run for more than one iteration.
5) If $i < m$ then $i++$, and go to Step 4.
6) Construct the Δ vector using Equation (2).
7) Sort the decision variables based on the magnitude of their corresponding delta value.
8) Stop if the maximum number of FEs is reached or go to Step 3 for the next *cycle*

We also developed a variant of DECC-D called DECC-DML that self-adapts the subcomponent sizes using the techniques used in DECC-ML[1]. DECC-DML differs from DECC-D in the way that it decomposes the decision vector. Instead of using a fixed subcomponent size in step 3, DECC-DML checks the fitness of the best individual and if there has been no improvement it will choose a different decomposer from a set of predetermined decomposers is called S. For this paper we used the following decomposers, $S = \{50, 100, 200, 250\}$.

IV. Empirical Results

We evaluated both DECC-D and DECC-DML on CEC'2008 benchmark functions the detailed description of which can be found in [11]. In order to further validate the results we also tested both DECC-D and DECC-DML on the new CEC'2010 benchmark functions [12]. The results of experiments on both set of benchmark functions are provided in Sections IV-A and IV-B respectively. We ran each algorithm for 25 independent runs for every function. The population size is set to 50 for all of the experiments.

A. Experiment Results on CEC'2008 Benchmark Functions

We tested our proposed algorithms with CEC'2008 function for 100, 500, and 1000 dimensions and the mean of the best fitness value over 25 runs was recorded. The performance of DECC-D and DECC-DML is compared with other algorithms in Tables I, II, III for 100, 500, and 1000 dimensions respectively. The maximum number of fitness evaluations(FEs) was calculated by the following formula, $FEs = 5000 \times D$, where D is the number of dimensions. In DECC-D the subcomponent size is set to 50, and for DECC-DML the following set of decomposers are used, $S = \{50, 100, 200, 250\}$.

As it can be seen from Tables I, and II DECC-DML outperforms MLCC on 6 out of 7 functions with 100, and 500 dimensions and on 1000 dimensions it outperforms MLCC on all the 7 functions(Table III). It is interesting to see that DECC-ML which completely relies on more frequent random grouping has the best performance, but it should be noted that several of CEC'2008 benchmark functions are separable and for those that are non-separable the degree of non-separability is unknown. By a closer look at Table III we can see that despite the better performance of DECC-ML the final mean fitness value is indeed very close to what is achieved by DECC-DML. Since the aim of this paper is to demonstrate the delta method as a new technique for grouping interacting variables, we leave further analysis of DECC-ML to another study.

TABLE I
Results of different algorithms over 100 dimensions(Averaged over 25 runs). Best results are highlighted in bold.

	DECC	DECC-ML	DECC-D	DECC-DML	MLCC
f_1	**2.7263e-29**	5.7254e-28	2.9283e-29	4.7379e-28	6.8212e-14
f_2	5.4471e+01	2.7974e-04	5.2479e+01	**2.4811e-04**	2.5262e+01
f_3	1.4244e+02	1.8871e+02	**1.4077e+02**	1.9233e+02	1.4984e+02
f_4	5.3370e+01	**0.0000e+00**	5.4444e+01	**0.0000e+00**	4.3883e-13
f_5	2.7589e-03	3.6415e-03	8.8753e-04	7.8858e-04	**3.4106e-14**
f_6	2.3646e-01	3.3822e-14	1.2270e-01	**3.1548e-14**	1.1141e-13
f_7	-9.9413e+02	-1.5476e+03	-9.8976e+02	**-1.5480e+03**	-1.5439e+03

TABLE II
Results of different algorithms over 500 dimensions(Averaged over 25 runs). Best results are highlighted in bold.

	DECC	DECC-ML	DECC-D	DECC-DML	MLCC
f_1	**8.0779e-30**	1.6688e-27	3.8370e-29	1.7117e-27	4.2974e-13
f_2	4.0904e+01	1.3396e+00	3.8009e+01	**1.0232e+00**	6.6663e+01
f_3	6.6822e+02	5.9341e+02	**5.6941e+02**	6.8292e+02	9.2466e+02
f_4	1.3114e+02	**0.0000e+00**	1.4631e+02	**0.0000e+00**	1.7933e-11
f_5	2.9584e-04	1.4788e-03	2.9584e-04	2.9584e-04	**2.1259e-13**
f_6	6.6507e-14	1.2818e-13	**5.9828e-14**	1.2051e-13	5.3433e-13
f_7	-5.5707e+03	**-7.4582e+03**	-4.7796e+03	-7.4579e+03	-7.4350e+03

TABLE III
Results of different algorithms over 1000 dimensions(Averaged over 25 runs). Best results are highlighted in bold.

	DECC	DECC-ML	DECC-D	DECC-DML	MLCC
f_1	1.2117e-29	5.1750e-28	**1.0097e-29**	3.3391e-27	8.4583e-13
f_2	4.2729e+01	**3.4272e+00**	3.8673e+01	5.81133e+00	1.0871e+02
f_3	1.2673e+03	**1.0990e+03**	1.1597e+03	1.22537e+03	1.7986e+03
f_4	2.4498e+02	**0.0000e+00**	2.7406e+02	**0.0000e+00**	1.3744e-10
f_5	2.9584e-04	9.8489e-04	**1.0392e-15**	1.4611e-15	4.1837e-13
f_6	1.3117e-13	2.5295e-13	**1.1866e-13**	2.2908e-13	1.0607e-12
f_7	-1.4339e+04	**-1.4757e+04**	-1.1035e+04	-1.4750e+04	-1.4703e+04

B. Experiment Results on CEC'2010 Benchmark Functions

In Table VI the best, worst, median, mean, and standard deviation are recorded. These information are recorded from different stages of evolution to demonstrate the convergence behavior of the algorithm. Figure 3 shows the convergence plot of DECC-D and DECC-DML for all of the variants of Rastrigin and Rosenbrock functions. We also compared the performance of DECC-D and DECC-DML with other algorithms such as DECC-G and MLCC as shown in Table VII. It can be seen that DECC-DML outperforms DECC-G on 14 out of 20 functions and outperforms MLCC on 12 out of 20 functions. This shows that the new delta method for grouping interacting variables is performing reasonably well.

In order to further investigate the performance of delta method we counted the number of interacting variables that DECC-DML managed to group in the first 50 variables regardless of the subcomponent sizes. Table IV shows the maximum number of interacting variables that were grouped for more than or equal to 2 cycles. Since functions $f_1 - f_3$ are separable it is meaningless to count the number of captured interacting variables for them. Functions f_{19}, f_{20} are completely non-separable so there is interaction between all of the variables so the number of captured interacting variables is not recorded for them. Let's consider f_4 as an example. The numbers in the table show that 46 interacting variables were captured by the delta method for 6 cycles. Since in DECC-DML the subcomponent size is self-adapted the maximum number of cycles varies from function to function, but it suffice to say that in our experiments this is always less than 7000 cycles for all of the benchmark function. Even by using 7000 cycles which is slightly higher than what is really used in our experiments, the probability of grouping 46 variables for at least two cycles will be virtually zero in case of random grouping. The entires in Table IV shows that the number of interacting variables that was captured by the delta method is reasonably high compared to random grouping, specially for functions $f_4 - f_8$. The entries for functions f_4,f_5,f_6,f_8 in Table VII confirms that high success rate of delta method in capturing interacting variables has direct relationship to its better performance compared to MLCC. Table V also shows the success rate for grouping at least 5 interacting variables. For example the success rate of f_{11} is approximately 53% which means in 53 out of 100 cycles at least 5 variables are captured amongst the first 50 variables. In order to compare this results with random grouping we calculated the probability of grouping 5 variables for more than 1 cycle using Equation (1). We use 7000 cycles for this example because none of the experiments with DECC-DML used more than 7000 cycles. It is also assumed that there are on average 10 different subcomponents so given $n = 1000$, $m = 10$, $N = 7000$ and $v = 5$, using Equation (1) we have:

$$P(X \geq 1) = 1 - P(X = 0) = 1 - \left(1 - \frac{1}{10^{5-1}}\right)^{7000} = 0.5034$$

which means that the probability of grouping 5 variables for at least two cycle is approximately 0.5. In the worst case of f_5 more than 5 interacting variables are grouped for exactly 182 cycles which is considerably higher than what is achievable using random grouping. It is also noteworthy that the results of Tables IV, and V are based on the number of interacting variables captured *only* in the first 50 variables. It is highly possible that more interacting variables are grouped within the next subcomponents. This is especially true for functions f_9-f_{18} where there are more than one group of interacting variables[12]. Another interesting pattern that can be seen from Tables IV and V is that delta method is highly effective for those set of non-separable functions that has only one group of interacting variables. Functions $f_4 - f_8$ have this property.

TABLE IV
MAXIMUM NUMBER OF INTERACTING VARIABLES CAPTURED BY DELTA METHOD FOR AT LEAST 2 CYCLES.

Function	f_4	f_5	f_6	f_7	f_8
# Captured Interacting Vars	46	32	50	31	49
# Cycles	6	3	506	4	4
Function	f_9	f_{10}	f_{11}	f_{12}	f_{13}
# Captured Interacting Vars	7	9	13	11	10
# Cycles	7	5	4	2	3
Function	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}
# Captured Interacting Vars	9	9	13	8	9
# Cycle	2	2	2	9	5

TABLE V
SUCCESS RATE OF DELTA METHOD ON GROUPING AT LEAST 5 INTERACTING VARIABLES. THE NUMBERS ARE DRAWN FROM THE SAME RUN AS IT WAS USED IN TABLE IV.

Function	f_4	f_5	f_6	f_7	f_8
Success Rate	99.87%	3.45%	99.76%	97.92%	99.84%
Function	f_9	f_{10}	f_{11}	f_{12}	f_{13}
Success Rate	6.98%	84.89%	53.85%	22.04%	21.71%
Function	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}
Success Rate	7.42%	9.07%	14.92%	10.97%	18.53%

TABLE VII
COMPARISON OF DIFFERENT ALGORITHMS ON CEC'2010 FUNCTIONS WITH 1000 DIMENSIONS. NUMBERS SHOW THE MEAN OF THE BEST FITNESS OVER 25 RUNS. BEST RESULTS ARE SHOWN IN BOLD.

Functions	DECC-DML	DECC-G	DECC-D	MLCC
f_1	1.925263e-25	2.93e-07	1.013417e-24	**1.53e-27**
f_2	2.169774e+02	1.31e+03	2.995242e+02	**5.57e-01**
f_3	**1.180922e-13**	1.39e+00	1.813305e-13	9.88e-13
f_4	**3.580284e+12**	1.70e+13	3.994117e+12	9.61e+12
f_5	2.985220e+08	**2.63e+08**	4.162337e+08	3.84e+08
f_6	**7.932774e+05**	4.96e+06	1.356873e+07	1.62e+07
f_7	1.387946e+08	1.63e+08	6.578934e+07	**6.89e+05**
f_8	**3.463122e+07**	6.44e+07	5.392069e+07	4.38e+07
f_9	**5.918405e+07**	3.21e+08	6.187354e+07	1.23e+08
f_{10}	1.246898e+04	1.06e+04	1.156625e+04	**3.43e+03**
f_{11}	**1.800515e-13**	2.34e+01	4.764118e+01	1.98e+02
f_{12}	3.795382e+06	8.93e+04	1.527193e+05	**3.49e+04**
f_{13}	1.144516e+03	5.12e+03	**9.867780e+02**	2.08e+03
f_{14}	**1.890322e+08**	8.08e+08	1.983536e+08	3.16e+08
f_{15}	1.540041e+04	1.22e+04	1.531490e+04	**7.11e+03**
f_{16}	**5.078991e-02**	7.66e+01	1.880495e+02	3.76e+02
f_{17}	6.536997e+06	2.87e+05	9.030164e+05	**1.59e+05**
f_{18}	2.472471e+03	2.46e+04	**2.123339e+03**	7.09e+03
f_{19}	1.586111e+07	**1.11e+06**	1.332509e+07	1.36e+06
f_{20}	**9.906186e+02**	4.06e+03	9.912724e+02	2.05e+03

V. CONCLUSION

In this paper we proposed a new technique called *delta method* for a more systematic way of grouping interacting variables of a non-separable problem into several subcomponents based on their sorted delta values. Delta values measure the averaged difference in a certain variable across the entire

TABLE VI

EXPERIMENT RESULTS OF CEC'2010 FUNCTIONS FOR 25 INDEPENDENT RUNS WITH 1000 DIMENSIONS.

1000D		f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
1.2e5	Best	2.28e+08	5.51e+03	8.22e+00	3.80e+13	1.43e+08	1.25e+06	2.65e+09	2.23e+09	4.09e+09	1.32e+04
	Median	2.85e+08	5.76e+03	9.71e+00	6.40e+13	2.85e+08	1.96e+06	5.50e+09	4.92e+09	4.91e+09	1.39e+04
	Worst	7.02e+08	5.96e+03	1.01e+01	1.20e+14	5.21e+08	2.00e+07	1.17e+10	1.35e+10	5.54e+09	1.45e+04
	Mean	4.09e+08	5.75e+03	9.51e+00	6.76e+13	3.00e+08	2.70e+06	5.97e+09	5.57e+09	4.89e+09	1.38e+04
	StDev	1.75e+08	1.35e+02	5.55e-01	2.02e+13	9.31e+07	3.62e+06	2.49e+09	2.56e+09	3.77e+08	3.24e+02
6.0e5	Best	6.95e+01	2.51e+03	1.06e-02	7.92e+12	1.42e+08	4.59e+01	3.14e+08	4.19e+07	2.82e+08	1.25e+04
	Median	4.63e+02	2.64e+03	1.83e-02	1.51e+13	2.85e+08	1.09e+02	5.42e+08	1.15e+08	3.85e+08	1.30e+04
	Worst	1.22e+03	2.78e+03	2.20e-02	3.29e+13	5.20e+08	1.98e+07	9.17e+08	2.38e+08	4.21e+08	1.36e+04
	Mean	6.02e+02	2.64e+03	1.81e-02	1.61e+13	2.99e+08	7.94e+05	5.84e+08	1.24e+08	3.73e+08	1.30e+04
	StDev	4.11e+02	5.88e+01	3.08e-03	6.19e+12	9.31e+07	3.97e+06	1.68e+08	5.40e+07	3.13e+07	2.93e+02
3.0e6	Best	9.05e-27	1.62e+02	1.10e-13	1.38e+12	1.42e+08	3.55e-09	7.09e+07	7.34e+05	4.51e+07	1.21e+04
	Median	1.22e-25	2.12e+02	1.14e-13	3.32e+12	2.85e+08	7.11e-09	1.23e+08	1.57e+07	5.97e+07	1.24e+04
	Worst	7.12e-25	2.94e+02	1.35e-13	6.89e+12	5.20e+08	1.98e+07	4.82e+08	1.21e+08	7.09e+07	1.30e+04
	Mean	1.93e-25	2.17e+02	1.18e-13	3.58e+12	2.99e+08	7.93e+05	1.39e+08	3.46e+07	5.92e+07	1.25e+04
	StDev	1.86e-25	2.98e+01	8.22e-15	1.54e+12	9.31e+07	3.97e+06	7.72e+07	3.56e+07	4.71e+06	2.66e+02

1000D		f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
1.2e5	Best	1.02e+02	4.07e+06	1.09e+08	1.26e+10	1.58e+04	3.22e+02	7.48e+06	1.56e+09	1.77e+07	2.04e+09
	Median	1.22e+02	4.68e+06	1.82e+08	1.37e+10	1.65e+04	3.73e+02	8.77e+06	3.30e+09	2.23e+07	3.93e+09
	Worst	1.70e+02	5.35e+06	3.72e+08	1.51e+10	1.73e+04	4.28e+02	1.01e+07	4.06e+09	2.72e+07	5.09e+09
	Mean	1.24e+02	4.70e+06	2.11e+08	1.37e+10	1.65e+04	3.75e+02	8.81e+06	3.08e+09	2.20e+07	3.84e+09
	StDev	1.38e+01	2.99e+05	9.68e+07	6.86e+08	3.61e+02	3.60e+01	6.86e+05	7.84e+08	2.36e+06	7.72e+08
6.0e5	Best	4.00e-01	3.64e+06	8.29e+02	9.82e+08	1.53e+04	3.46e+00	6.50e+06	5.64e+03	1.54e+07	1.43e+03
	Median	7.09e-01	4.22e+06	1.71e+03	1.18e+09	1.59e+04	8.65e+00	7.29e+06	1.48e+04	1.84e+07	1.67e+03
	Worst	1.72e+00	4.65e+06	1.47e+04	1.29e+09	1.67e+04	4.28e+02	7.99e+06	3.96e+04	2.46e+07	2.02e+03
	Mean	7.66e-01	4.19e+06	3.15e+03	1.17e+09	1.59e+04	4.47e+01	7.27e+06	1.74e+04	1.87e+07	1.69e+03
	StDev	2.81e-01	2.18e+05	3.09e+03	8.20e+07	3.63e+02	1.16e+02	3.77e+05	8.26e+03	1.99e+06	1.58e+02
3.0e6	Best	1.63e-13	3.46e+06	6.19e+02	1.54e+08	1.48e+04	2.74e-13	5.65e+06	1.64e+03	1.30e+07	9.69e+02
	Median	1.78e-13	3.81e+06	1.06e+03	1.89e+08	1.53e+04	3.20e-13	6.55e+06	2.21e+03	1.59e+07	9.75e+02
	Worst	2.03e-13	4.11e+06	2.09e+03	2.22e+08	1.62e+04	1.27e+00	7.63e+06	7.52e+03	2.16e+07	1.10e+03
	Mean	1.80e-13	3.80e+06	1.14e+03	1.89e+08	1.54e+04	5.08e-02	6.54e+06	2.47e+03	1.59e+07	9.91e+02
	StDev	9.88e-15	1.50e+05	4.31e+02	1.49e+07	3.59e+02	2.54e-01	4.63e+05	1.18e+03	1.72e+06	3.51e+01

population. Experimental results confirmed that this new technique is capable of grouping interacting variables. For the class of non-separable problems with a single group of interacting variables, delta method have shown great performance and managed to group more than 80% of interacting variables.

Delta method seems to be a promising technique for large scale non-separable optimization problems but this new technique is still in its infancy and further research is required in order to fully understand its underlying mechanics. For example it is not clear why DECC-DML is less efficient on non-separable functions with more than one group of rotated variables. Further investigation is also required in order to understand the reason for good performance of DECC-ML. As it was briefly mentioned earlier this might be due to the fact that most CEC'2008 benchmark functions are mostly separable and DECC-DML might pay off when applied to CEC'2010 functions. This calls for a separate study for comparing the performance of DECC-DML and DECC-ML on CEC'2010 test suite.

ACKNOWLEDGMENT

The authors would like to thank Mr. Zhenyu Yang for providing us with the source code of DECC-G and SaNSDE algorithms. This work was partially supported by an EPSRC grant (No. EP/G002339/1) on "Cooperatively Coevolving Particle Swarms for Large Scale Optimisation".

REFERENCES

[1] M. N. Omidvar, X. Li, Z. Yang, and X. Yao, "Cooperative co-evolution for large scale optimization through more frequent random grouping," in *Proc. of IEEE World Congress on Computational Intelligence*, 2010, in press.

[2] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences*, vol. 178, pp. 2986–2999, August 2008.

[3] T. Ray and X. Yao, "A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning," in *Proc. of IEEE Congress on Evolutionary Computation*, May 2009, pp. 983–989.

[4] R. Salomon, "Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions - a survey of some theoretical and practical aspects of genetic algorithms," *BioSystems*, vol. 39, pp. 263–278, 1995.

[5] M. A. Potter and K. A. D. Jong, "A cooperative coevolutionary approach to function optimization," in *Proc. of the Third Conference on Parallel Problem Solving from Nature*, vol. 2, 1994, pp. 249–257.

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

[7] D. Sofge, K. D. Jong, and A. Schultz, "A blended population approach to cooperative coevolution fordecomposition of complex problems," in *Proc. of IEEE World Congress on Computational Intelligence*, 2002, pp. 413–418.

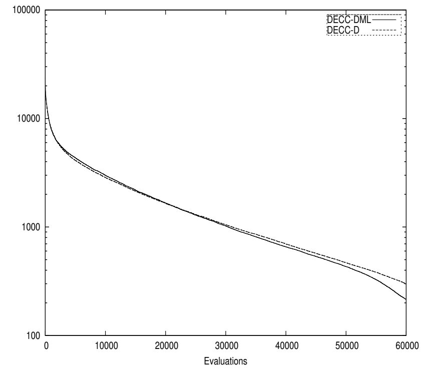
[8] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation 8 (3)*, pp. 225–239, 2004.

[9] Y. Shi, H. Teng, , and Z. Li, "Cooperative co-evolutionary differential evolution for function optimization," in *Proc. of the First International Conference on Natural Computation*, 2005, pp. 1080–1088.

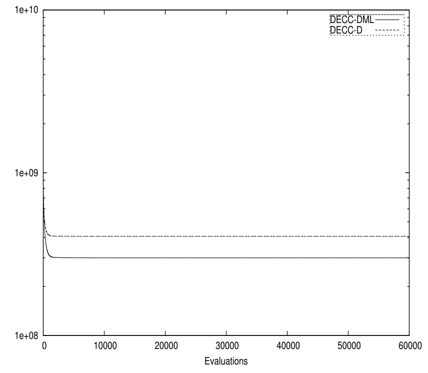
[10] Z. Yang, K. Tang, and X. Yao, "Multilevel cooperative coevolution for large scale optimization," in *Proc. of IEEE World Congress on Computational Intelligence*, June 2008, pp. 1663–1670.

[11] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, , and Z. Yang, "Benchmark functions for the cec'2008 special session and competition on large scale global optimization," Nature Inspired Computation and Applications Laboratory, USTC, China, Tech. Rep., 2007, http://nical.ustc.edu.cn/cec08ss.php.

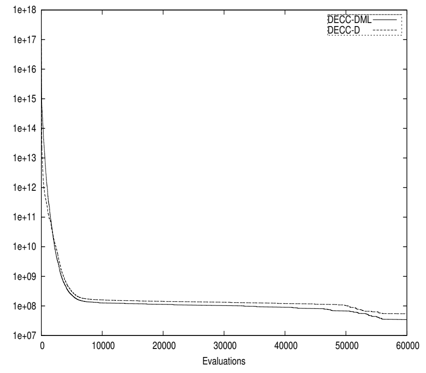
[12] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, "Benchmark functions for the cec'2010 special session and competition on large-scale global optimization," NICAL, USTC, China, Tech. Rep., 2009, http://nical.ustc.edu.cn/cec10ss.php.



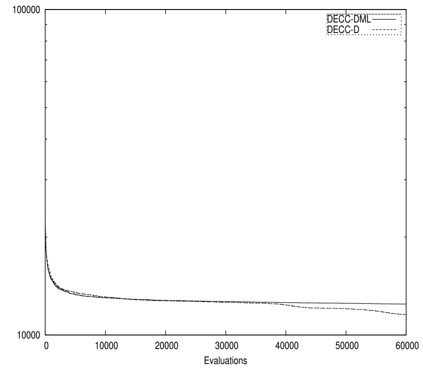
(a) f_2



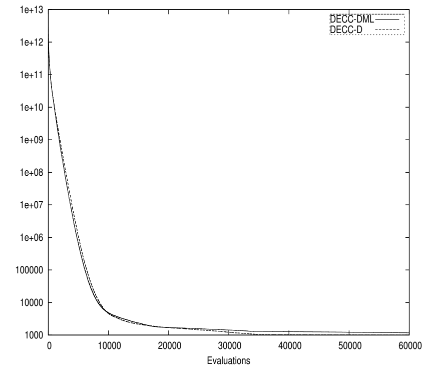
(b) f_5



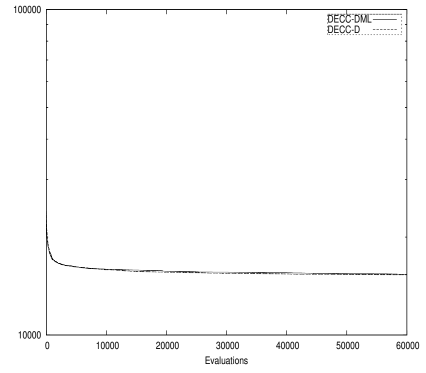
(c) f_8



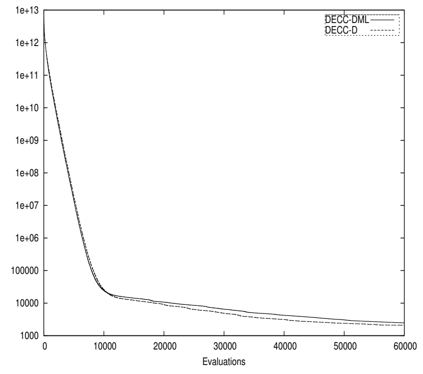
(d) f_{10}



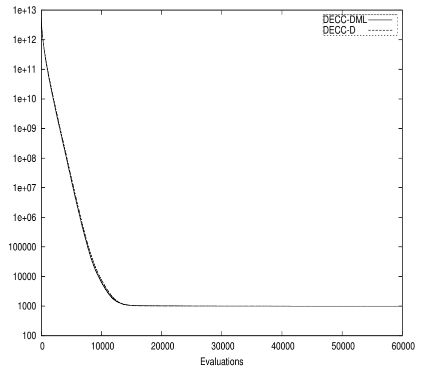
(e) f_{13}



(f) f_{15}



(g) f_{18}



(h) f_{20}

Fig. 3. Convergence plots of $f_2, f_5, f_8, f_{10}, f_{13}, f_{15}, f_{18}$, and f_{20}(All variants of Rastrigin and Rosenbrock functions) for DECC-D and DECC-DML. Each point on the graph is the average over 25 independent runs.