Tackling High Dimensional Nonseparable Optimization Problems By Cooperatively Coevolving Particle Swarms

Xiaodong Li, *Senior Member, IEEE*, Xin Yao, *Fellow, IEEE*

***Abstract*— This paper attempts to address the question of scaling up Particle Swarm Optimization (PSO) algorithms to high dimensional optimization problems. We present a cooperative coevolving PSO (CCPSO) algorithm incorporating random grouping and adaptive weighting, two techniques that have been shown to be effective for handling high dimensional nonseparable problems. The proposed CCPSO algorithms outperformed a previously developed coevolving PSO algorithm on nonseparable functions of 30 dimensions. Furthermore, the scalability of the proposed algorithm to high dimensional nonseparable problems (of up to 1000 dimensions) is examined and compared with two existing coevolving Differential Evolution (DE) algorithms, and new insights are obtained. Our experimental results show the proposed CCPSO algorithms can perform reasonably well with only a small number of evaluations. The results also suggest that both the random grouping and adaptive weighting schemes are viable approaches that can be generalized to other evolutionary optimization methods.**

I. Introduction

Stochastic algorithms such as Evolutionary Algorithms (EAs) and Particle Swarm Optimization (PSO) algorithms have been shown to be effective optimisation techniques [1]. However, their performance often deteriorates rapidly as the dimensionality of the problem increases. A natural approach to tackle high dimensional optimisation problems is to adopt a *divide-and-conquer* strategy. An early work by Potter and De Jong on a cooperative coevolutionary algorithm (CCEA) [2] provides a promising approach for decomposing a high dimensional problem and tackling its subcomponents simultaneously. By cooperatively coevolving multiple EA subpopulations, each dealing with a subproblem of a lower dimensionality, we can obtain an overall solution derived from combinations of sub-solutions evolved from individual sub-populations. Clearly, the effectiveness of such CCEAs depends heavily on the decomposition strategies used. The classical CCEAs [2] performed poorly on nonseparable problems, because the inter-independencies among different variables could not be captured well enough by the algorithms. Recent improvements [3] achieved only limited success in dealing with variable inter-independencies in cooperative coevolution on low dimensional problems. Generally speaking, existing CCEAs or CCPSO algorithms performed poorly on nonseparable problems with 100 or more real-valued variables effectively.

Xiaodong Li is with the School of Computer Science and IT, RMIT University, VIC 3001, Melbourne, Australia (phone: +61 3 99259585; fax: +61 3 96621617; email: xiaodong.li@rmit.edu.au).

Xin Yao is with the School of Computer Science, The University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K. (phone: +44 121 4143747; fax: +44 121 4142799; email: x.yao@cs.bham.ac.uk).

PSO is an efficient optimization algorithm which has gained increasing popularity in recent years [1]. Classical PSO algorithms have been shown to perform well on low dimensional problems, however often poorly on high dimensional problems [4], [5]. In [3], van den Bergh and Engelbrecht made a first attempt to apply Potter's CCGA to PSO in order to address the issue of scaling up PSO to higher dimensional problems. Two cooperative PSO models, CPSO-S_K and CPSO-H_K, were developed. Unfortunately, these two models were only tested on problems with up to 30 dimensions. The question remains on how well these CCPSO models scale with problems of 100 or more dimensions.

More recently, a decomposition strategy based on random grouping was presented by Yang *et al.* [6], [7]. Without prior knowledge of the non-separability of a problem, it was shown that random grouping increases the probability of two interacting variables being allocated to the same subcomponent, thereby making it possible to optimize these interacting variables in the same subcomponent, rather than across different subcomponents. This strategy was embedded in a differential evolution (DE) model, and was evaluated on problems up to 1000 dimensions. In addition, an adaptive weighting scheme was developed to further fine-tune the solutions produced by the coevolutionary DE model. The DE using the cooperative coevolutionary framework that integrate these two schemes (so called DECC-G) outperformed existing EAs especially on high dimensional nonseparable problems [7]. However, it is unclear whether these two schemes can be generalized to other optimization techniques such as PSO in improving their performances. Along this line of thinking, this paper attempts to address the following questions:

- Can we develop more effective CCPSO algorithms especially for solving high dimensional nonseparable optimisation problems?
- In light of the recent works by Yang *et al.* [6], [7], can we generalize the two proposed strategies, random grouping and adaptive weighting, to the CCPSO algorithms?
- Can we further improve the performance of the CCPSO proposed by van den Bergh and Engelbrecht [3]?

This paper is organized as follows. Section II presents an overview of existing CCEAs, especially those tested for handling nonseparable or high dimensional problems. Section III provides the rationale on why PSO is an appropriate choice for building a CC model, as well as the previous work on CCPSO algorithms, which our newly proposed CCPSO is

978-1-4244-2959-2/09/$25.00 ©

built upon. Section IV describes two techniques that have proven to be useful in improving the performance of an cooperative coevolutionary DE model, and furthermore how they can be incorporated into a CCPSO model. Section V describes the experimental setup, followed by Section VI presenting experimental results and analysis. Finally Section VII gives the concluding remarks.

II. Cooperative Coevolution

The first CCEA for function optimization, called CCGA, was proposed by Potter and De Jong [2], where six test functions of up to 30 dimensions were used in the experiments. However, no attempt was made in using the cooperative coevolution (CC) framework for higher dimensional problems. More recently, the idea of using CC in optimization has attracted much attention and was incorporated into several algorithms, including evolutionary programming [8], evolution strategies [9], PSO [3] and DE [10], [6], [7].

In their original CCGA, Potter and De Jong [2] decomposed a problem into several smaller components, each being evolved by a separate GA subpopulation. When a subpopulation is being evolved, all of the other subpopulations are held fixed. The subpopulations are each evolved in a round-robin fashion. For a function optimization problem of n variables, Potter and De Jong [2] decomposed it into n subcomponents, corresponding to n subpopulations, one for each variable. The fitness of a subpopulation member is determined by the n-dimensional vector formed by this member and selected members from other subpopulations. In a way, the fitness of a subpopulation member is assessed by how well it "cooperates" with other subpopulations. Two models of cooperation were examined. In the first model CCGA-1, the fitness of a subpopulation member is computed by combining it with the current best members of other subpopulations. It was found that CCGA-1 performed significantly better than a conventional GA on separable problems, but much worse on non-separable problems. To improve CCGA's performance on non-separable problems, CCGA-2 was proposed where members were randomly selected from other subpopulations in the fitness evaluation. On a 2-dimensional Rosenbrock function, CCGA-2 was shown to perform better than CCGA-1. In summary, Potter and De Jong's original study [2] showed great potentials of the CC framework for function optimization. However, the CCGA framework was tested only on problems with up to 30 dimensions.

Liu *et al.* [8] applied the CC framework to their Fast Evolutionary Programming (FEP) algorithms. The new algorithm FEPCC (FEP with Cooperative Coevolution) was able to tackle benchmark functions with 100 to 1000 real-valued variables. However, for one of the non-separable functions, FEPCC performed poorly and trapped in a local optimum, confirming the deficiency of handling variable interactions in Potter and De Jong's decomposition strategy [2].

van den Bergh and Engelbrecht [3] first introduced the CC framework to PSO. Two cooperative PSO algorithms, CPSO-S_K and CPSO-H_K, were developed. CPSO-S_K adopts the same framework as that of Potter's CCGA, except that it allows a vector to be split into K subcomponents, instead of each single dimension being a separate subcomponent. CPSO-H_K is a hybrid combining both a standard PSO with the CPSO-S_K. These two CPSO algorithms were tested on some benchmark problems with up to 30 dimensions. Some rotated test functions with variable interactions were used. Their results showed that correlation among variables in such problems reduces the effectiveness of the two CPSO algorithms. However, no new decomposition strategies were proposed especially for handling high dimensional nonseparable problems.

New decomposition strategies were proposed and investigated for DE with CC [10], [6], [7]. A splitting-in-half strategy was proposed by Shi *et al.* [10], which decomposed the search space into two subcomponents, each was evolved by a separate subpopulation. Clearly, this strategy does not scale up very well and loses its effectiveness quickly when the number of dimensions increases. Yang *et al.* [6], [7] reported some promising results of using a decomposition strategy based on *random grouping* with DE on high dimensional nonseparable problems (up to 1000 real-valued variables). Although the proposed algorithm, DECC-G [7], outperformed all other algorithms significantly, the solutions found were still some distances away from the true global optimal value. There is ample room for further improvement. More analysis is needed to understand how well different decomposition strategies capture interdependencies among different variables.

III. Particle Swarm Optimization

PSO is modelled on an abstract framework of "collective intelligence" in social animals [1], [4]. In PSO, individual particles of a swarm represent potential solutions, which "fly" through the problem search space seeking the optimal solution. These particles broadcast their current positions to neighbouring particles. Previously identified "good positions" are then used by the swarm as a starting point for further search, where individual particles adjust their current positions and velocities.

A defining characteristic of PSO is its fast convergent behaviour and inherent adaptability, especially when compared to conventional EAs. Theoretical analysis of PSO [4] has shown that particles in a swarm can switch between an exploratory (with large search step sizes) and an exploitative (with smaller search step sizes) mode, responding adaptively to the shape of the fitness landscape. This characteristic makes PSO an ideal candidate for incorporating the CC framework for problems of high complexity and dimensionality. For more information on PSO, please refer to [4].

A. CPSO-S_K and CPSO-H_K

In [3], van den Bergh and Engelbrecht developed two cooperative PSO algorithms. In the first CPSO variant, CPSO-S_K, they adopted the original decomposition strategy from Potter and De Jong [2], but permitting a vector to be split into K subcomponents, each corresponding to a swarm of s-dimensions (where $n = Ks$). Algorithm 1 illustrates the

Create and initialize K swarms, each with s dimensions (where $n = Ks$); The j-th swarm is denoted as $P_j, j \in [1..K]$;
repeat
  for *each swarm* $j \in [1..K]$ **do**
    for *each particle* $i \in [1..s]$ **do**
      if $f(\mathbf{b}(j, P_j.x_i)) < f(\mathbf{b}(j, P_j.y_i))$ **then**
      $P_j.y_i = P_j.x_i$;
      if $f(\mathbf{b}(j, P_j.y_i)) < f(\mathbf{b}(j, P_j.\hat{y}))$ **then**
      $P_j.\hat{y} = P_j.y_i$;
    end
    Perform velocity and position updates for each particle in P_j;
  end
until *termination criterion is met*;

Algorithm 1: The pseudocode of the CPSO-S$_K$ algorithm. $P_j.x_i$ denotes the current position of the i-th particle of the j-th swarm, whereas $P_j.y_i$ is the personal best of the i-th particle of the j-th swarm. The j-th of the K swarms has a global best particle $P_j.\hat{y}$. The function $\mathbf{b}(j, z)$ returns a vector $(P_1.\hat{y}, P_2.\hat{y}, \ldots, P_{j-1}.\hat{y}, z, P_{j+1}.\hat{y}, \ldots, P_K.\hat{y})$.

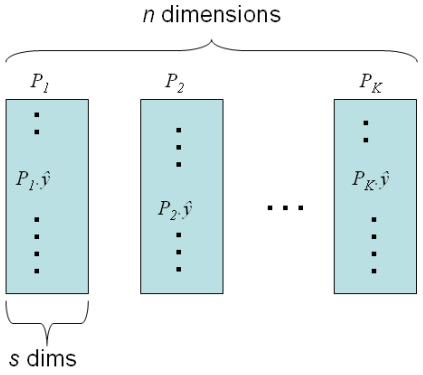


Fig. 1. The concatenation of $P_1.\hat{y}, P_2.\hat{y}, \ldots, P_K.\hat{y}$ constitutes $\hat{y}$.

working of CPSO-S$_K$ [3]. In order to evaluate the fitness of a particle in a swarm, a *context vector* $\hat{y}$ is constructed, which is a concatenation of all global best particles from all K swarms (as shown in Figure 1). The evaluation of the i-th particle in the j-th swarm is done by calling the function $\mathbf{b}(j, P_j.x_i)$ which returns an n-dimensional vector consisting of $\hat{y}$ with its j-th component replaced by $P_j.x_i$. The idea is to evaluate how well $P_j.x_i$ "cooperate" with the best individuals from all other swarms.

Note that if K equals n, CPSO-S$_K$ operates the same way as Potter's CCGA-1, where n subpopulations of 1-D vectors are coevolved.

In their second variant, CPSO-H$_K$, both CPSO-S$_K$ and a standard PSO are used in an alternating fashion, with CPSO-S$_K$ executed for one iteration, followed by the standard PSO in the next iteration. Information exchange between CPSO-S$_K$ and the standard PSO was allowed so that the best solution found so far can be shared. More specifically, after an iteration of CPSO-S$_K$, the context vector $\hat{y}$ is used to replace a randomly chosen particle in the standard PSO part, and similarly, if the standard PSO finds a new global best particle, this vector will replace the global best particle in the CPSO-S$_K$ part.

Both CPSO-S$_K$ and CPSO-H$_K$ were tested on functions of up to 30 dimensions only, however it is unclear how well the performances of CPSO-S$_K$ and CPSO-H$_K$ scale with functions of even higher dimensions. In this paper, we will extend the CPSO-S$_K$ by incorporating the random grouping and adaptive weighting schemes (as described in the following sections) on problems up to 1000 dimensions. The results of these new CCPSO variants will be also compared with CPSO-S$_K$ and CPSO-H$_K$.

IV. CCPSO WITH RANDOM GROUPING AND ADAPTIVE WEIGHTING

This section describes how the *random grouping* based decomposition method and the *adaptive weighting* scheme, which were recently proposed by Yang, *et al.* [7], can be incorporated into the CCPSO algorithm.

A. Random grouping

In CPSO-S$_K$ [3], the n-dimensional search space is decomposed into K subcomponents, each corresponding to a swarm of s-dimensions (where $n = Ks$). However, the s variables in any given swarm remain in the same swarm over the course of optimization. Since it is not always known in advance how these K subcomponents are related, it is possible that such a static grouping method places some interacting variables into different subcomponents. Because CCEAs work better if interacting variables are placed within the same subcomponent, instead of across different subcomponents, this static grouping method in CPSO-S$_K$ (as well as in other existing CCEAs) will encounter difficulty in dealing with nonseparable problems.

One method to alleviate this problem is to dynamically change the grouping structure as suggested in [7]. Here, if we randomly decompose the n-dimensional object vector into K subcomponents at each iteration, i.e., we construct each of the K subcomponents by randomly selecting s-dimensions from the n-dimensional object vector, the probability of placing two interacting variables into the same subcomponent becomes higher, over an increasing number of iterations. For example, for a problem of 1000 dimensions, if $K = 10$ (hence we know $s = n/K = 100$), the probability of placing two variables into the same subcomponent at one iteration is $p = \frac{\binom{10}{1}}{10} = 0.1$. If we run the algorithm for 50 iterations, 50 executions of random groupings will occur. The probability of optimizing the two variables in the same subcomponent for at least one iteration follows a binomial probability distribution [7], and can be computed as follows:

$$
\begin{aligned}
P(x \geq 1) &= p(1) + p(2) + \cdots + p(50) \\
&= 1 - p(0) \\
&= 1 - \binom{50}{0}(0.1)^0(1-0.1)^{50} \\
&= 0.9948
\end{aligned}
$$

where x denotes the number of observed "successes" of placing two variables in the same subcomponent over the 50 trials. This suggests the random grouping strategy should help when there are some variable interactions present in a problem.

B. Adaptive weighting

The purpose of employing adaptive weighting is to further fine-tune the solutions evolved by CPSO-S_K (which may or may not use random grouping). In this scheme, each of the K subcomponents (as shown in Figure 1) can be associated with a weight, hence a weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_K)$ can be constructed for all K subcomponents. Specifically, each of the n variables is weighted by its corresponding weight value of the vector $(\underline{w_1, \ldots, w_1}, \underline{w_2, \ldots, w_2}, \ldots, \underline{w_K, \ldots, w_K})$, where each 'underlined' subvector denotes s identical weights for a subcomponent of s variables. For a solution produced by CPSO-S_K, we can then evolve a population of weight vectors associated with this solution, hoping to improve the solution further. In this paper, we choose to optimize the weight vectors associated with the *context vector* $\hat{y}$, which is a concatenation of all global best particles from all K swarms. If $\hat{y}$ gets improved, it will be communicated back to CPSO-S_K, so that the new $\hat{y}$ can be used to guide the particles in each swarm in the future executions of CPSO-S_K. Since usually $K \ll n$, optimizing the weight vectors is a much less costly computational task than optimizing the n-dimensions of the original problem.

In order to keep a weighted $\hat{y}$ always within the feasible search space, we must set appropriate lower and upper bounds for the range of possible weight values. This can be achieved by first identifying the feasible weight interval for each dimension of $\hat{y}$, and then taking the intersection of all such weight intervals. For example, if we have a 3-dimensional context vector $\hat{y} = (1, 2, -3)$, with each dimension in the range [-10,10]. The 3 weight intervals are [-10/1, 10/1], [-10/2, 10/2], and [10/-3, -10/-3] respectively. The intersection of them is [-10/3, 10/3]. For any weight value in [-10/3, 10/3], the weighted $\hat{y}$ should always stay in the range [-10, 10].

Algorithm 2 shows the working of the newly proposed CCPSO-S_K with *random grouping* and *adaptive weighting* schemes, which we build upon the previous CPSO-S_K. For the adaptive weighting scheme, we may use a simple optimization method such as PSO to optimize the weight vectors.

```
Create and initialize K swarms, each with s
dimensions (where n = Ks); The j-th swarm is
denoted as P_j, j ∈ [1..K];
repeat
    // CCPSO-S_K module employing random grouping
    Randomly permutate all n dimension indices;
    Construct K swarms, each with s dimensions;
    for each swarm j ∈ [1..K] do
        for each particle i ∈ [1..s] do
            if f(b(j, P_j.x_i)) < f(b(j, P_j.y_i)) then
            P_j.y_i = P_j.x_i;
            if f(b(j, P_j.y_i)) < f(b(j, P_j.ŷ)) then
            P_j.ŷ = P_j.y_i;
        end
        Perform velocity and position updates for each
        particle in P_j;
    end
    // Additional module employing adaptive weighting
    Create and initialize a population of weight
    vectors, which are associated with ŷ;
    Use a PSO to evolve the weight vectors to
    improve ŷ for a few iterations;
    If improved, ŷ is updated for future iterations;
until termination criterion is met;
```

Algorithm 2: The pseudocode of the proposed CCPSO-S_K employing random grouping and adaptive weighting.

V. Experimental setup

We chose to use the same set of test functions as those used by van den Bergh and Engelbrecht [3] in order to make the comparisons of our results with theirs easier (as shown in Table I), except that for f_0, a more commonly used generalized Rosenbrock function was used. These functions were further rotated to introduce variable interactions, thereby making them nonseparable [1]. Rotations are performed in the decision space, on each plane using a random uniform rotation matrix [11], [12]. A new random uniform rotation matrix is generated for each individual run for the purpose of an unbiased assessment.

Experiments were carried out in two stages. In the first stage, experiments on the 5 test functions of 30 dimensions were run for a maximum of 200,000 evaluations. We tried to use the same experimental setup as those in [3]. All experiments were run 50 times, and the average of the best object vector fitness values (i.e., $\hat{y}$) was recorded. The number of dimensions for each swarm s was set to 5, hence K was 6 (since $K = n/s$). Thus there are 6 swarms. The population size for each swarm was set to 20. The Type 1" constricted PSO was adopted [4] for both CCPSO-S_K and optimizing weight vectors. φ_1 and φ_2 were set to 2.05, while χ set to 0.7298. Whenever the adaptive weighing scheme

[1]Note that f_0 and f_4 are already nonseparable functions.

TABLE I
TEST FUNCTIONS.

Test function	**Range**	f_{min}
Generalized Rosenbrock: $f_0(\mathbf{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30, 30]^n$	0
Quadric: $f_1(\mathbf{x}) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^n$	0
Ackley: $f_2(\mathbf{x}) = -20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e$	$[-30, 30]^n$	0
Generalized Rastrigin: $f_3(\mathbf{x}) = \sum_{i=1}^{n} (x_i^2 - 10cos(2\pi x_i) + 10)$	$[-5.12, 5.12]^n$	0
Generalized Griewank: $f_4(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]^n$	0

was invoked, the constricted PSO was allowed to run for 10 iterations for optimizing the weight vectors.

The following CCPSO-S_K variants were used for comparisons:

- CCPSO-S_K: Identical to the CPSO-S_K described in [3], where K is the number of swarms used;
- CCPSO-S_K-rg: CCPSO-S_K employing *random grouping* only;
- CCPSO-S_K-aw: CCPSO-S_K employing *adaptive weighting* only;
- CCPSO-S_K-rg-aw: CCPSO-S_K employing both random grouping and adaptive weighting.

The results of these CCPSO-S_K variants were compared with those of CPSO-H_K (as reported in [3]), which was considered to be the best CCPSO performing algorithm especially when dealing with rotated functions.

In the second stage, experiments on the same functions of 500 and 1000 dimensions were carried out. Experiments were again run for a maximum of 200,000 evaluations. Since these experiments were rather time consuming, only 25 runs were executed, and the averaged results were recorded. s was set to 100, hence K was 5 and 10 for 500 and 1000 dimensional functions respectively. The population size for each swarm was set to 50.

VI. RESULTS

This section presents the experimental results and analysis. We first compared the performances of the 4 CCPSO-S_K variants and the results of CPSO-H_K [3] on the test functions of 30 dimensions. The 4 CCPSO-S_K variants were then tested on functions of 500 and 1000 dimensions, and their results were compared with two existing DECC variants, DECC-I [6] and DECC-G [7].

Unequal variance t-tests [13] were conducted to see whether the performances of CCPSO-S_K-rg-aw and CCPSO-S_K-aw are different. The value of the two tailed t-test is significant at $\alpha = 0.05$. This will allow us to see more clearly the effect of using a combination of random grouping and adaptive weighting or just adaptive weighting alone.

A. 30 dimensional functions

Table II shows the results on test functions of 30 dimensions. As to overall performances, CCPSO-S_6-aw and CCPSO-S_6-rg-aw clearly outperformed the other CCPSO-S_6 variants, and the previously developed CPSO-H_6 [3]. The only exception is f_1 Quadric function, where CPSO-H_6 is the winner. However, since f_1 is a separable function, it is probably not surprising that even CPSO-H_6 and CCPSO-S_6-aw (where random grouping is not employed) performed reasonably well on this function. It is noticeable that CCPSO-S_6-rg (i.e., only using random grouping) performed poorly on the 30 dimensional functions. However, CCPSO-S_6-aw (i.e., only using adaptive weighting) is more effective, as shown in Table II. It is evident that the adaptive weighting scheme was able to effectively fine-tune the best solution, and feeded it back to CCPSO-S_6, and subsequently further improving the performance of the CCPSO-S_6 algorithm. It can be noted that when a function is rotated, an algorithm tends to suffer from some degree of performance degradation. Among these variants, the performances of CCPSO-S_6-rg-aw suffers the least from the rotation effect. For f_2 and f_3, and their rotated versions f_{2r} and f_{3r}, identical results were obtained, while for the remaining functions, rotation only degraded the performance of CCPSO-S_6-rg-aw slightly. In contrast, the level of performance degradation tended to be rather large for the other algorithms. In particular, for CPSO-H_6, the performance degradation is substantial on f_{1r} the rotated Quadric function. Though random grouping alone is ineffective, combining it with adaptive weighting (i.e., CCPSO-S_6-rg-aw) was able to provide the best performances on f_2, f_3, f_4, and their rotated versions, except f_{4r}. However, CCPSO-S_6-rg-aw did not perform as well as CCPSO-S_6-aw on f_0, f_1 and their rotated versions.

Figures 2 and 3 show the convergence behaviours of the 4 CCPSO-S_6 variants. Among all variants, CCPSO-S_6-rg-aw is clearly least sensitive to the effect of function rotation across all test functions. Although CCPSO-S_6-aw performed reasonably well, its performance deteriorates rapidly when encountering a rotated test function. CCPSO-S_6-rg-aw which employs both random grouping and adaptive weighting was able to alleviate this problem to some extent.

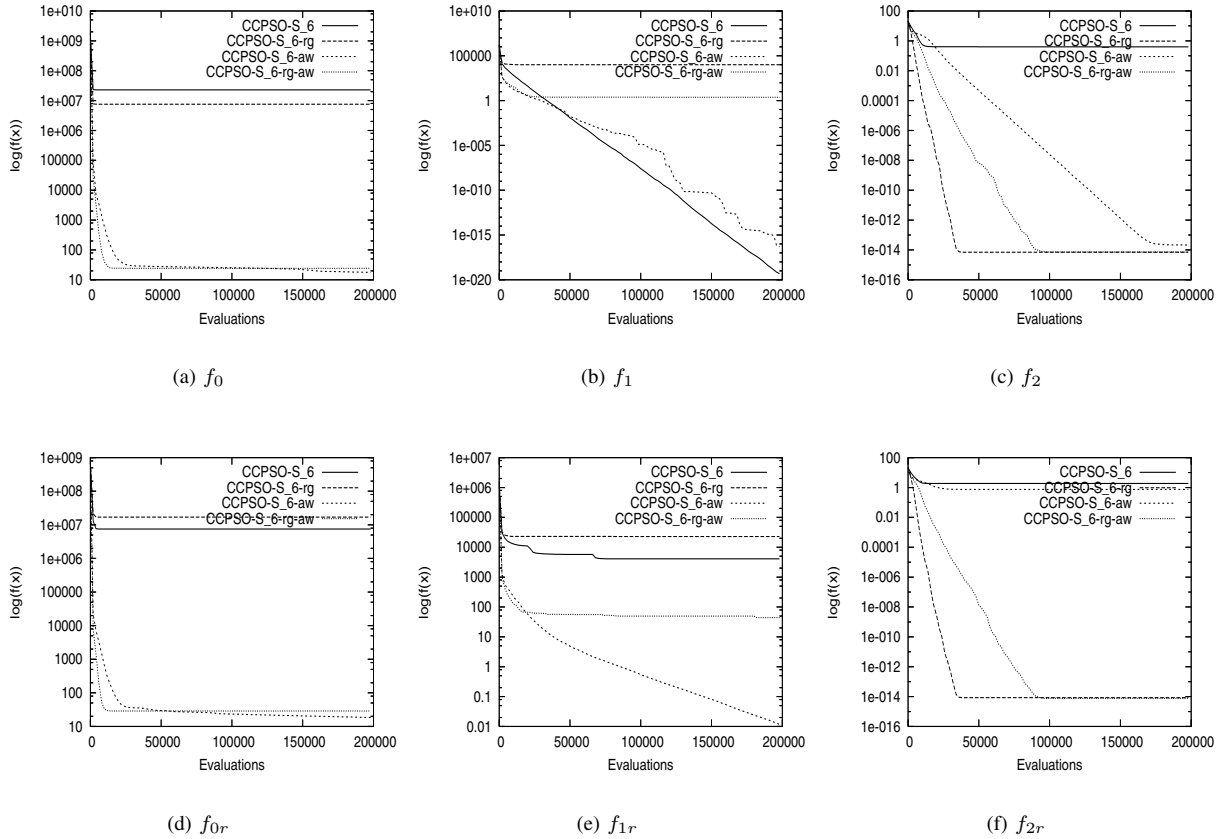
B. 500 and 1000 dimensional functions

When tested on functions of 500 and 1000 dimensions, the effect of using only random grouping or a combination of random grouping and adaptive weighting becomes more pronounced than on 30 dimensions. Because rotation was performed on each plane of the decision space of a rotated test function, making each variable more or less interacting with every other variable (see section V), for higher dimensional

TABLE II

BEST FUNCTION VALUES ON TEST FUNCTIONS OF 30 DIMENSIONS. THE LAST COLUMN GIVES THE p-VALUE FROM THE UNEQUAL VARIANCE t-TEST BETWEEN CCPSO-S_6-AW AND CCPSO-S_6-RG-AW.

fnc	CCPSO-S_6	CCPSO-S_6-rg	CCPSO-S_6-aw	CCPSO-S_6-rg-aw	CPSO-H_6	p-value
f_0	2.28E+07 (5.83E+07)	7.58E+06 (3.77E+07)	**1.76E+01 (3.99E+00)**	2.41E+01 (1.06E+01)	N/A	0.00015
f_{0r}	7.48E+06 (3.73E+07)	1.70E+07 (5.24E+07)	**1.84E+01 (1.31E+00)**	2.89E+01 (5.81E-02)	N/A	1.46E-46
f_1	2.70E-20 (7.03E-20)	9.95E+03 (2.58E+04)	4.33E-17 (3.03E-16)	1.29E+00 (5.33E+00)	**1.40E-29 (1.15E-29)**	0.09333
f_{1r}	4.08E+03 (2.02E+04)	2.29E+04 (2.05E+04)	**1.05E-02 (2.04E-02)**	5.07E+01 (1.25E+02)	1.03E+03 (5.24E+02)	0.00607
f_2	4.00E-01 (2.83E+00)	7.05E-15 (6.26E-15)	2.13E-14 (4.95E-15)	**4.44E-16 (3.49E-31)**	2.73E-12 (2.03E-12)	4.53E-33
f_{2r}	1.87E+00 (2.72E+00)	8.54E-15 (8.25E-15)	7.43E-01 (9.21E-01)	**4.44E-16 (3.49E-31)**	1.51E-12 (6.83E-13)	6.76E-07
f_3	2.22E+01 (6.64E+00)	8.16E+00 (5.74E+01)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	7.78E-01 (1.87E-01)	N/A
f_{3r}	1.28E+02 (2.83E+01)	1.99E+02 (6.46E+01)	1.13E+00 (4.29E+00)	**0.00E+00 (0.00E+00)**	5.41E+01 (4.63E+00)	0.0674
f_4	2.63E-02 (2.32E-02)	1.99E+01 (9.92E+01)	1.39E-02 (2.62E-02)	**2.59E-03 (1.83E-02)**	5.24E-02 (1.19E-02)	0.01411
f_{4r}	1.80E+01 (8.98E+01)	9.85E+00 (6.95E+01)	**1.12E-02 (1.26E-02)**	4.99E-02 (2.00E-01)	4.06E-02 (1.03E-02)	0.17918



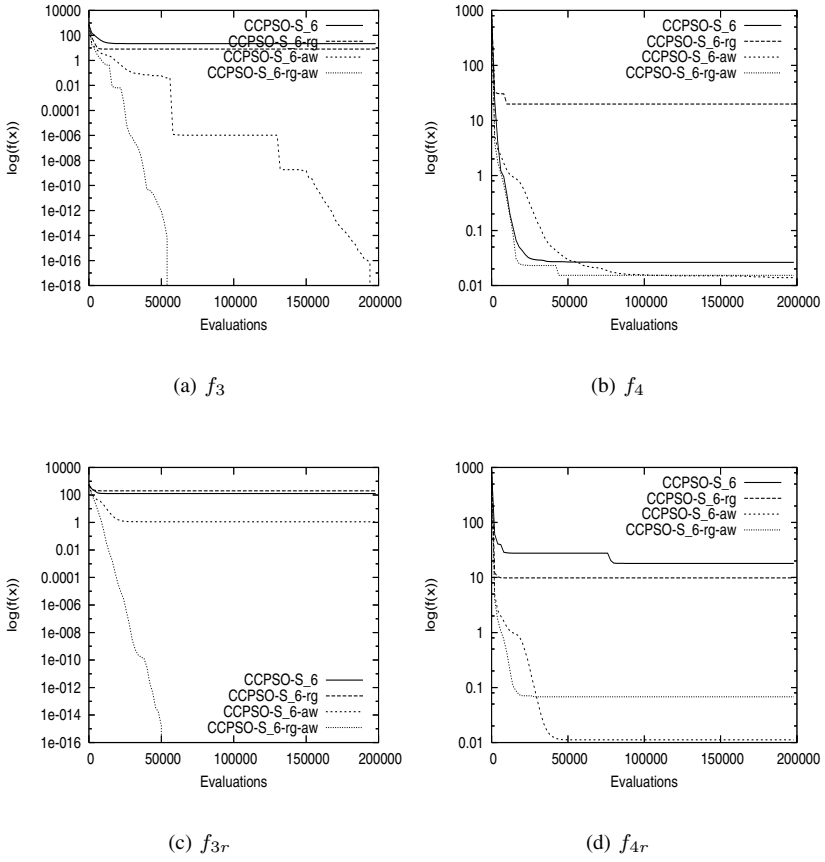
(a) f_0 (b) f_1 (c) f_2

(d) f_{0r} (e) f_{1r} (f) f_{2r}

Fig. 2. The fitness values of $\hat{y}$ for the 4 CCPSO-S_6 variants on f_0, f_1, f_2 of 30 dimensions, and their rotated versions.

functions, this means there are more interacting variables (with strong or weak interactions). For each subcomponent, since we have $s = 100$ for the 500 and 1000 dimensional functions, but only $s = 6$ for the 30 dimensional functions. Hence there are far more interacting variables captured in the same subcomponent for the experiments on the 500 and 1000 dimensions than the 30 dimensional case. As a result, employing random grouping has a more significant impact on the experiments for the 500 and 1000 dimensional functions than 30 dimensions.

As shown in Table III, using random grouping (i.e., CCPSO-S_5-rg) is much better than not (i.e., CCPSO-S_5). This contrasts with the results on 30 dimensions, where the differences are not so clear. In fact, CCPSO-S_5-rg performed equally well or better than CCPSO-S_5-rg-aw on 5 out of 10 functions on 500 dimensions. Both CCPSO-S_5-rg and CCPSO-S_5-rg-aw are significantly better than DECC-I as reported in [6]. Compared with the more recently developed DECC-G [7], CCPSO-S_5-rg-aw performed better on f_0 and equally well on f_3, though worst on others. Even CCPSO-S_5-rg still performed better than DECC-G on f_0 and f_4. Bear in mind that both DECC-I and DECC-G used a maximum



(a) f_3 (b) f_4

(c) f_{3r} (d) f_{4r}

Fig. 3. The fitness values of $\hat{y}$ for the 4 CCPSO-S_6 variants on f_3, f_4 of 30 dimensions, and their rotated versions.

of 2.5E+06 evaluations, which is 12.5 times that of CCPSO-S_5-rg and CCPSO-S_5-rg-aw though.

The performance of CCPSO-S_5-rg-aw is significantly better than CCPSO-S_5-aw on f_{0r}, f_2, f_{2r}, f_{3r}, f_4 and f_{4r}. For the remaining functions, its performances are not significantly different from that of CCPSO-S_5-aw. It is remarkable that CCPSO-S_5-rg-aw found the perfect minimum for f_3 and its rotated function f_{3r}. This contrasts to the poor performance by the classic PSO on f_3 of only 30 and 100 dimensions [5]. Comparing the results of CCPSO-S_5-rg-aw and CCPSO-S_5-rg, it can be noted that adding adaptive weighting is helpful, since CCPSO-S_5-rg-aw is substantially better than CCPSO-S_5-rg on f_1, f_{1r}, f_2, f_3, and f_{3r}, and equally well on f_{0r}, but slightly worse on others.

For functions of 1000 dimensions, DECC-G is the overall best performer (but it used more than 10 times of evaluations than CCPSO-S_{10}-rg and CCPSO-S_{10}-rg-aw). The performance of CCPSO-S_{10}-rg-aw is significantly better than CCPSO-S_{10}-aw on f_3, f_4 and f_{4r}, but no significant differences are observed for the other functions. It is also worth noting that CCPSO-S_{10}-rg is better than CCPSO-S_{10}-rg-aw on 4 functions.

Overall, the results on high dimensional nonseparable functions provide strong empirical evidences that employing random grouping is highly desirable. Furthermore, combining random grouping with adaptive weighting can provide additional performance gain than employing adaptive weighting alone.

VII. Conclusions

Traditional PSO algorithms have been known to perform poorly especially on high dimensional and nonseparable problems [5]. To combat this issue, this paper investigates the potential of using a new decomposition method (i.e., random grouping) and a solution fine-tuning method (i.e., adaptive weighting) in a cooperative coevolutionary PSO (CCPSO) algorithm. Among the CCPSO variants proposed, CCPSO-S_K-rg-aw, which employs a hybrid of random grouping and adaptive weighting, has shown to be robust to functions that have variable interactions. The proposed CCPSO-S_K-rg-aw outperformed a previously developed CCPSO algorithms [3], on a number of nonseparable 30-dimensional functions. More importantly, the performances of CCPSO-S_K-rg-aw on functions of 500 and 1000 dimensions suggest that the cooperative coevolutoinary approach has a great potential to further improve the scalability of PSO algorithms in tackling high dimensional nonseparable optimization problems. This paper will be a starting point of hopefully many more research works on this topic.

Compared with two existing algorithms in handling 500

TABLE III

BEST FUNCTION VALUES ON TEST FUNCTIONS OF 500 DIMENSIONS. THE LAST COLUMN GIVES THE p-VALUE FROM THE UNEQUAL VARIANCE t-TEST BETWEEN CCPSO-S_5-AW AND CCPSO-S_5-RG-AW.

fnc	CCPSO-S_5	CCPSO-S_5-rg	CCPSO-S_5-aw	CCPSO-S_5-rg-aw	DECC-I	DECC-G	p-value
f_0	1.73E+07 (7.19E+07)	**4.15E+02 (1.85E+02)**	1.97E+03 (4.02E+03)	4.54E+02 (1.37E+02)	9.65E+04	4.92E+02	0.07182
f_{0r}	1.72E+06 (6.93E+05)	**4.99E+02 (6.11E-02)**	6.76E+02 (2.82E+02)	**4.99E+02 (7.58E-02)**	N/A	N/A	0.00433
f_1	3.51E+06 (7.71E+06)	3.58E+05 (7.60E+05)	1.57E+00 (5.32E+00)	5.35E+00 (2.41E+01)	1.75E+08	**6.17E-25**	0.45024
f_{1r}	1.59E+06 (3.41E+06)	4.11E+06 (4.44E+06)	2.01E+02 (4.06E+02)	**7.66E+01 (1.57E+02)**	N/A	N/A	0.16143
f_2	9.43E+00 (2.46E+00)	1.34E-09 (6.70E-09)	8.01E-02 (1.10E-01)	4.04E-10 (1.15E-09)	2.58E+03	**9.13E-14**	0.00131
f_{2r}	1.58E+01 (1.27E+00)	**8.69E-15 (3.51E-15)**	2.27E-01 (2.16E-01)	1.67E-06 (8.35E-06)	N/A	N/A	2.2E-05
f_3	2.78E+03 (1.52E+03)	2.05E+02 (2.63E+02)	1.79E+00 (3.93E+00)	**0.00E+00 (0.00E+00)**	2.22E+03	**0.00E+00**	0.03160
f_{3r}	3.40E+03 (3.04E+02)	1.11E+03 (2.01E+03)	7.28E+00 (9.40E+00)	**0.00E+00 (0.00E+00)**	N/A	N/A	0.00073
f_4	6.93E+01 (3.61E+01)	**3.55E-17 (5.29E-17)**	5.64E-01 (4.21E-01)	5.19E-04 (2.59E-03)	N/A	4.40E-16	6.35E-07
f_{4r}	7.12E+01 (3.27E+01)	**4.44E-18 (2.22E-17)**	4.63E-01 (4.93E-01)	1.11E-14 (4.13E-14)	N/A	N/A	9.12E-05

TABLE IV

BEST FUNCTION VALUES ON TEST FUNCTIONS OF 1000 DIMENSIONS. THE LAST COLUMN GIVES THE p-VALUE FROM THE UNEQUAL VARIANCE t-TEST BETWEEN CCPSO-S_{10}-AW AND CCPSO-S_{10}-RG-AW.

fnc	CCPSO-S_{10}	CCPSO-S_{10}-rg	CCPSO-S_{10}-aw	CCPSO-S_{10}-rg-aw	DECC-I	DECC-G	p-value
f_0	2.30E+09 (5.10E+09)	4.74E+08 (2.37E+09)	1.20E+04 (2.68E+04)	1.16E+03 (8.52E+02)	1.98E+05	**9.87E+02**	0.05502
f_{0r}	2.72E+09 (6.60E+09)	**9.98E+02 (6.14E-02)**	1.74E+03 (1.22E+03)	1.87E+03 (3.76E+03)	N/A	N/A	0.87742
f_1	1.38E+07 (2.66E+07)	1.46E+07 (2.75E+07)	3.49E+02 (1.56E+03)	5.88E+02 (1.65E+03)	3.85E+08	**3.71E-23**	0.60033
f_{1r}	2.82E+07 (3.24E+07)	1.48E+07 (9.38E+06)	**3.55E+03 (4.80E+03)**	6.90E+03 (2.20E+04)	N/A	N/A	0.46358
f_2	1.47E+01 (2.80E+00)	8.00E-01 (4.00E+00)	1.91E-01 (2.60E-01)	7.36E-02 (2.18E-01)	6.85E+03	**2.22E-13**	0.08942
f_{2r}	1.97E+01 (2.95E-01)	**1.49E-10 (5.46E-10)**	5.46E-01 (7.19E-01)	2.43E-01 (5.35E-01)	N/A	N/A	0.09792
f_3	8.05E+03 (4.96E+03)	1.19E+03 (1.06E+03)	8.89E+00 (1.34E+01)	3.60E-03 (1.79E-02)	4.59E+03	**3.55E-16**	0.00295
f_{3r}	8.60E+03 (3.66E+02)	8.44E+03 (4.46E+03)	6.31E+01 (1.77E+02)	**5.69E-01 (2.65E+00)**	N/A	N/A	0.08971
f_4	2.12E+03 (4.60E+03)	**2.80E-16 (4.79E-16)**	1.28E+00 (1.07E+00)	6.03E-02 (2.66E-01)	N/A	1.01E-15	6.84E-06
f_{4r}	2.18E+03 (4.38E+03)	**7.97E-13 (3.64E-12)**	1.43E+00 (8.62E-01)	1.26E-01 (4.46E-01)	N/A	N/A	7.25E-08

and 1000 dimensional functions, CCPSO-S_K-rg-aw is shown to have relatively fast convergence, capable of providing reasonably good performance with only a small number of evaluations. Our results also suggest both random grouping and adaptive weighting are viable techniques that can be generalized across to various evolutionary optimization methods.

In future, we will be interested in examining techniques that allow us to detect variable interactions that exist in a problem, and subsequently making use of this knowledge in the decomposition procedure to more effectively identify and group interacting variables while keeping independent variables in separate groups. We will be also interested in studying what characteristics of a non-separable problem are challenging to CCPSO. Such insight will be very useful in guiding the design of novel CCPSO algorithms in the future.

ACKNOWLEDGMENT

The authors would like to thank Zhenyu Yang for his very helpful discussion on DECC-G. This research is partially supported by an EPSRC grant (No. EP/G002339/1) on "Cooperatively Coevolving Particle Swarms for Large Scale Optimisation". Part of the work was done while the first author was visiting Birmingham.

REFERENCES

[1] J. Kennedy and R. Eberhart, *Swarm Intelligence*. Morgan Kaufmann, 2001.

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

[3] F. van den Bergh and A. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[4] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 58–73, 2002.

[5] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proceedings of the 2004 Congress on Evolutionary Computation (CEC'04)*, vol. 2, pp. 1980–1987, IEEE, 2004.

[6] Z. Yang, K. Tang, and X. Yao, "Differential evolution for high-dimensional function optimization," in *Proceedings of the 2007 Congress on Evolutionary Computation*, pp. 3523–3530, IEEE, 2007.

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

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

[9] D. Sofge, K. D. Jong, and A. Schultz, "A blended population approach to cooperative coevolution for decomposition of complex problems," in *Proc. of the Congress on Evolutionary Computation*, pp. 413–418, 2002.

[10] 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*, pp. 1080–1088, 2005.

[11] R. Salomon, "Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions," *BioSystems*, vol. 39, pp. 263–278, 1996.

[12] A. Iorio and X. Li, "Rotated test problems for assessing the performance of multiobjective optimization algorithms," in *Proceeding of Genetic and Evolutionary Computation Conference 2006 (GECCO'06)* (e. a. M. Keijzer, ed.), pp. 683–690, ACM Press, 2006.

[13] B. Moser, G. Stevens, and C. Watts, "The two-sample t-test versus satterwaite's approximate f test," *Commun. Stat. Theory Methodol.*, vol. 18, pp. 3963–3975, 1989.