Cooperatively Coevolving Particle Swarms for Large Scale Optimization

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

***Abstract*—This paper presents a new cooperative coevolving particle swarm optimization (CCPSO) algorithm in an attempt to address the issue of scaling up particle swarm optimization (PSO) algorithms in solving large-scale optimization problems (up to 2000 real-valued variables). The proposed CCPSO2 builds on the success of an early CCPSO that employs an effective variable grouping technique *random grouping*. CCPSO2 adopts a new PSO position update rule that relies on Cauchy and Gaussian distributions to sample new points in the search space, and a scheme to dynamically determine the coevolving subcomponent sizes of the variables. On high-dimensional problems (ranging from 100 to 2000 variables), the performance of CCPSO2 compared favorably against a state-of-the-art evolutionary algorithm sep-CMA-ES, two existing PSO algorithms, and a cooperative coevolving differential evolution algorithm. In particular, CCPSO2 performed significantly better than sep-CMA-ES and two existing PSO algorithms on more complex multimodal problems (which more closely resemble real-world problems), though not as well as the existing algorithms on unimodal functions. Our experimental results and analysis suggest that CCPSO2 is a highly competitive optimization algorithm for solving large-scale and complex multimodal optimization problems.**

***Index Terms*—Cooperative coevolution, evolutionary algorithms, large-scale optimization, particle swarm optimization, swarm intelligence.**

I. Introduction

STOCHASTIC algorithms such as evolutionary algorithms (EAs) and particle swarm optimization (PSO) algorithms have been shown to be effective optimization techniques [1]. However, their performance often deteriorates rapidly as the dimensionality of the problem increases. Nevertheless, many real-world problems involve optimization of a large number of variables. For example, in shape optimization a large number of shape design variables is often used to represent complex shapes, such as turbine blades [2], aircraft wings [3], and heat exchangers [4]. Existing EAs are often ill-equipped in handling this class of problems. To meet such a demand, research into designing EAs that are able to tackle large-scale optimization problems has recently gained momentum [5], [6].

PSO is notorious for being prone to premature convergence. For example, in a comparative study where several widely used stochastic algorithms such as differential evolution (DE), EA, and PSO were evaluated [7], PSO was shown to perform very poorly, as the dimensionality of a problem increases. This perception of PSO's inability to handle high-dimensional problems seems to be widely held [7], [8].

A natural approach to tackle high-dimensional optimization problems is to adopt a *divide-and-conquer* strategy. An early work on a cooperative coevolutionary algorithm (CCEA) by Potter and Jong [9] provides a promising approach for decomposing a high-dimensional problem, and tackling its subcomponents individually. 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 subsolutions, which are evolved from individual subpopulations. Clearly, the effectiveness of such CCEAs depends heavily on the decomposition strategies used. Classical CCEAs [9] performed poorly on nonseparable problems, because the interdependencies among different variables could not be captured well enough by the algorithms. Generally speaking, existing CCEAs still performed poorly on nonseparable problems with 100 or more real-valued variables [10].

An early attempt to apply Potter's CC model to PSO was made by Van den Bergh and Engelbrecht [8], where two cooperative PSO models, CPSO-S_K and CPSO-H_K, were developed. However, these two models were only tested on functions of up to 30 dimensions [8] and 190 dimensions [11]. The question remains on how well these CCPSO models scale with problems of significantly larger dimensions.

Recent studies by Yang *et al.* [10], [12] suggest a new decomposition strategy based on *random grouping*. Without prior knowledge of the nonseparability 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. An *adaptive weighting* scheme was also introduced to further fine-tune the solutions produced by the CCEA [10]. Inspired by these recent works, a CCPSO integrating the *random grouping* and *adaptive weighting* schemes was developed, and demonstrated great promise in scaling up PSO on high-dimensional nonseparable problems [13]. This

Manuscript received January 5, 2010; revised September 29, 2010, May 17, 2010 and January 4, 2011; accepted January 12, 2011. Date of publication June 23, 2011; date of current version March 30, 2012. This work was supported by EPSRC, under Grant EP/G002339/1, which funded the first author's two trips to Birmingham, U.K., as a Visiting Research Fellow in 2008 and 2009.

X. Li is with the School of Computer Science and Information Technology, Royal Melbourne Institute of Technology, Melbourne, VIC 3001, Australia (e-mail: xiaodong.li@rmit.edu.au).

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

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2011.2112662

CCPSO outperformed the previously proposed CPSO-H_K [8] on 30-D separable and nonseparable functions. CCPSO was also shown to perform reasonably well on these functions of up to 1000 dimensions. Nevertheless, our latest paper on the CCEAs [14] revealed that it is actually more beneficial to apply random grouping more frequently than using the adaptive weighting scheme.

Building on our preliminary work on CCPSO [13] and our new findings on random grouping [14], this paper aims to demonstrate convincingly that a CC approach is an effective *divide-and-conquer* strategy and can be utilized to help scaling up PSO's performance for solving problems with a large number of variables. The proposed CCPSO2 described in this paper enhances the previously proposed CCPSO substantially. CCPSO2 differs from CCPSO in the following aspects.

1) A new PSO model using Cauchy and Gaussian distributions for sampling around the personal best and the neighborhood best (respectively), is proposed. This PSO model using an *lbest* ring topology shows an improved search capability compared with existing PSO models (see Section VI-A).
2) In the context of a CC framework, an *lbest* ring topology is used to define a local neighbourhood instead of the standard *gbest* model, to improve performance, especially on multimodal optimization problems.
3) A new strategy for updating personal bests and the global best in the context of a CC framework is developed (see Section IV-D).
4) The adaptive weighting scheme is removed, since our recent work [14] shows that it is more cost effective to apply random grouping more frequently instead.
5) A new adaptive scheme is used to dynamically determine the subcomponent sizes for random grouping during a run, hence removing the need to specify this parameter.
6) A comprehensive study comparing CCPSO2 with another state-of-the-art global optimization algorithm sep-CMA-ES on high-dimensional functions is provided. In addition, comparisons were made with two existing PSO algorithms and a CC DE which were specially designed for handling large-scale optimization problems.
7) Results on functions of up to 2000 dimensions.

Three major benefits of using CCPSO2 include: CCPSO2 employs the Cauchy and Gaussian-based update rules in conjunction with an *lbest* topology hence its search capability is enhanced; CCPSO2 is more reliable and robust to use—a user does not have to specify the subcomponent size, since it is adaptively chosen from a set. Furthermore, our results show that CCPSO2 performs significantly better than the state-of-the-art algorithm sep-CMA-ES and two existing PSO models on complex multimodal functions of up to 2000 dimensions.

The rest of this paper is organized as follows. Section II presents an overview of existing CCEAs, specifically, those tested for handling nonseparable or high-dimensional problems. Random grouping as a novel decomposition method for CCEAs is also described. Section III provides the rationale on why PSO is an appropriate choice for constructing a CC model, as well as a previous study on CCPSO models, upon which our newly proposed CCPSO2 is built. Section IV introduces our new CCPSO2 algorithm, including the Cauchy and Gaussian PSO (CGPSO) adopted and a new scheme that allows dynamically changing subcomponent sizes during a run. 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

A. Early Work

The first CCEA for function optimization, called CCGA, was proposed by Potter and Jong [9], where the algorithm was empirically evaluated on six test functions of up to 30 dimensions. However, no attempt was made in using the cooperative coevolution (CC) framework on 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 [15], evolution strategies [16], PSO [8], and DE [10], [12], [17].

In their original CCGA, Potter and Jong [9] decomposed a problem into several smaller subcomponents, each evolved by a separate GA subpopulation. As each subpopulation is evolved, the remaining subpopulations are held fixed. The subpopulations are evolved in a round-robin fashion. For a function optimization problem of n variables, Potter and Jong [9] decomposed the problem 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 nonseparable problems. To improve CCGA's performance on nonseparable problems, CCGA-2 was proposed where members were randomly selected from other subpopulations in the fitness evaluation. On a 2-D Rosenbrock function, CCGA-2 was shown to perform better than CCGA-1. In summary, Potter and Jong's original study [9] demonstrated the efficacy of the CC framework applied to function optimization. However, the CCGA framework was tested only on problems of up to 30 dimensions.

Liu *et al.* [15] applied the CC framework to their fast evolutionary programming (FEP) algorithms. The new algorithm FEP with CC (FEPCC) was able to optimize benchmark functions with 100 to 1000 real-valued variables. However, for one of the nonseparable functions, FEPCC performed poorly and was trapped in a local optimum, confirming the deficiency of handling variable interactions in Potter and Jong's decomposition strategy [9].

Van den Bergh and Engelbrecht [8] 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 subcomponent consisting of a single dimension. CPSO-H_K is a hybrid approach combining both a standard PSO with the CPSO-S_K. These two CPSO algorithms were tested on some benchmark problems of up to 30 dimensions (and 190 dimensions in [11]). Some rotated test functions with variable interactions were also used. Their results demonstrated that correlation among variables in such problems reduces the effectiveness of the two CPSO algorithms. However, no new decomposition strategies were proposed for handling high dimensional nonseparable problems. A similar cooperative approach was also adopted in [18] but was implemented for bacterial foraging optimization.

New decomposition strategies were proposed and investigated for DE with CC [10], [12], [17], [19]. A splitting-in-half strategy was proposed by Shi *et al.* [17], which decomposed the search space into two subcomponents, each evolved by a separate subpopulation. Clearly, this strategy does not scale up very well and loses its effectiveness quickly when the number of dimensions becomes very large. Yang *et al.* [10], [12] proposed a decomposition strategy based on *random grouping* of variables, and applied it to a CC DE, on high-dimensional nonseparable problems with up to 1000 real-valued variables. The proposed algorithms, DECC-G [10] and subsequently MLCC [19], outperformed several existing algorithms significantly. This random grouping strategy represents an important step forward in handling nonseparable high-dimensional problems, and will be incorporated into our proposed CCPSO algorithm. We will describe this *random grouping* technique in detail in the next section.

B. Random Grouping of Variables

In CPSO-S_K [8], the n-dimensional search space is decomposed into K subcomponents, each corresponding to a swarm of s-dimensions (where $n = K * s$). 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 for any given problem, it is likely 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 is likely to encounter difficulty in dealing with nonseparable problems.

One method to alleviate this problem is to dynamically change the grouping structure [10]. We call this method *random grouping*, which is the simplest dynamic grouping method and does not assume any prior knowledge of the problem to be optimized. 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 in one iteration is $p = \frac{\frac{10}{1}}{10} = 0.1$. 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, and can be computed as follows:

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

where x denotes the number of observed "successes" of placing two variables in the same subcomponent over the 50 trials; $p(1)$ denotes the probability of having one such "success" over 50 iterations, and similarly $p(2)$ being the probability of having two such "successes," and so on. This suggests the random grouping strategy should help when there are some variable interactions present in a problem. Our recent paper [14] further generalizes the above probability calculation to cases where more than two interacting variables are present.

III. Particle Swarm Optimization

PSO is modeled on an abstract framework of "collective intelligence" in social animals [1], [20]. 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 neighboring 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 distinct characteristic of PSO is its fast convergent behavior and inherent adaptability, especially when compared to conventional EAs. Theoretical analysis of PSO [20] 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 to be incorporated into the CC framework for handling problems of high complexity and dimensionality.

In a canonical PSO, the velocity of each particle is modified iteratively by its *personal best* position (i.e., the position giving the best fitness value so far) and the *global best* position (i.e., the position of the best-fit particle from the entire swarm). As a result, each particle searches around a region defined by its *personal best* position and *global best* position. Let $\mathbf{v}_i$ denote the velocity of the ith particle in the swarm, $\mathbf{x}_i$ its position, $\mathbf{y}_i$ its personal best position, and $\hat{\mathbf{y}}$ the global best position from the entire swarm. Each dth dimension of $\mathbf{v}_i$ and $\mathbf{x}_i$ of the ith particle in the swarm are updated according to the two equations [20] as follows:

$$\begin{aligned} v_{i,d}(t+1) &= \chi(v_{i,d}(t) + c1r1_{i,d}(t)(y_{i,d}(t) - x_{i,d}(t)) + \\ &\quad c2r2_{i,d}(t)(\hat{y}_d(t) - x_{i,d}(t))) \qquad (1) \\ x_{i,d}(t+1) &= x_{i,d}(t) + v_{i,d}(t+1) \qquad (2) \end{aligned}$$

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

```
Create and initialize K swarms, each with s dimensions
(where n = K * s); The jth swarm is denoted as
P_j, j ∈ [1..K];
repeat
    for each swarm j ∈ [1..K] do
        for each particle i ∈ [1..swarmSize] 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 using (1)
        and (2) for each particle in P_j;
    end
until termination criterion is met;
```

for all $i \in \{1, \ldots, swarmSize\}$ and $d \in \{1, \ldots, n\}$ (where $swarmSize$ is the population size of the swarm and n is the number of dimensions). $c1$ and $c2$ are *acceleration coefficients*. $r1_{i,d}$ and $r2_{i,d}$ are two random values independently and uniformly generated from the range $[0, 1]$. A constriction coefficient χ is used to prevent each particle from exploring too far away in the search space, since χ applies a dampening effect to the oscillation size of a particle over time. This "Type 1" constricted PSO suggested by Clerc and Kennedy is often used with χ set to 0.7298, and $c1$ and $c2$ set to 2.05 [20].

A. CPSO-S_K and CPSO-H_K

Van den Bergh and Engelbrecht [8] developed two cooperative PSO algorithms. In the first CPSO variant, CPSO-S_K, they adopted the original decomposition strategy from Potter and Jong [9], but allowing a vector to be split into K subcomponents, each corresponding to a swarm of s-dimensions (where $n = K * s$). Algorithm 1 illustrates CPSO-S_K [8]. In order to evaluate the fitness of a particle in a swarm, a *context vector* $\hat{\mathbf{y}}$ is constructed, which is a concatenation of all global best particles from all K swarms (as shown in Fig. 1). The evaluation of the ith particle in the jth swarm is done by calling the function $\mathbf{b}(j, P_j.\mathbf{x}_i)$ which returns an n-dimensional vector consisting of $\hat{\mathbf{y}}$ with its jth component replaced by $P_j.\mathbf{x}_i$. The idea is to evaluate how well $P_j.\mathbf{x}_i$ "cooperates" 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 manner, 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. To be specific, after an iteration of CPSO-S_K, the context vector $\hat{\mathbf{y}}$ is used to replace a randomly chosen particle in the standard PSO. This is followed by one iteration of standard PSO, which may yield a new global best solution. This new best solution can be then used to update the subvectors of a randomly chosen particle from CPSO-S_K.

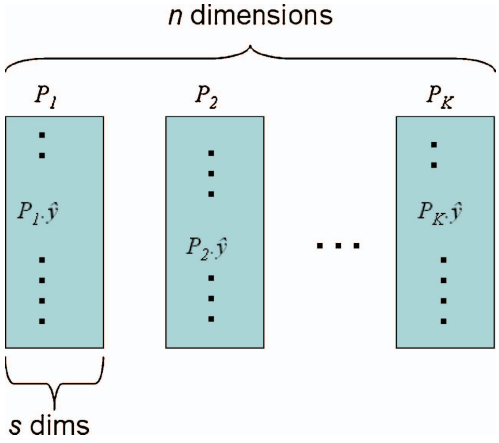

Fig. 1. Concatenation of $P_1.\hat{\mathbf{y}}, P_2.\hat{\mathbf{y}}, \ldots, P_K.\hat{\mathbf{y}}$ constitutes $\hat{\mathbf{y}}$.

Both CPSO-S_K and CPSO-H_K were tested on functions of up to 30 dimensions [8], however, it is unclear how well the performances of CPSO-S_K and CPSO-H_K scale with functions of higher dimensions. In our preliminary study of cooperatively coevolving PSO [13], the previously proposed CCPSO, which employed both *random grouping* and *adaptive weighting* schemes, not only outperformed CPSO-S_K on several 30-D functions, but also showed promising performance on these functions of up to 1000 dimensions.

IV. New CCPSO2 Algorithm

This paper proposes CCPSO2, which builds upon the previously proposed CCPSO [13]. CCPSO2 incorporates several new schemes to improve the performance and reliability of CCPSO. First, we adopt a PSO that does not use the velocity term, but instead, employ Cauchy and Gaussian distributions to generate the next particle positions. Second, we use an *lbest* ring topology to define the local neighborhood for each particle in order to slow down convergence and maintain better population diversity. Third, instead of using a fixed group (or subcomponent) size throughout a run for the *random grouping* mechanism, a different group size can be randomly chosen from a set at each iteration. There is clear evidence that the application of *random grouping* together with adaptively choosing subcomponent sizes contributed to the marked improvements of CCEAs [14], [19].

In the following sections, we will first review several studies which we have drawn upon to propose the new *lbest* PSO model using Cauchy and Gaussian distributions for sampling, then describe a simple scheme for dynamically changing the group size when random grouping is applied, and finally how these are put together to form the new CCPSO2 algorithm.

A. Related Studies

1) *Gaussian-Based PSO:* Two most commonly used PSO variants are probably the *inertia weight* PSO and *constricted*

PSO. An early study [20] suggested that these two are equivalent to each other. Other studies suggested that a Gaussian distribution could be used in the PSO position update rule. For example Kennedy proposed a Bare-bones PSO [21], where each dimension of the new position of a particle is randomly selected from a Gaussian distribution with the mean being the average of $y_{i,d}(t)$ and $\hat{y}_d(t)$ and the standard deviation σ being the distance between $y_{i,d}(t)$ and $\hat{y}_d(t)$ as follows:

$$x_{i,d}(t+1) = \mathcal{N}\left(\frac{y_{i,d}(t)+\hat{y}_d(t)}{2}, |y_{i,d}(t)-\hat{y}_d(t)|\right). \quad (3)$$

Note that there is no velocity term used in (3). The new particle position is simply generated via the Gaussian distribution. A comparative study on PSO variants employing Gaussian distribution was provided in [22], including a Lévy distribution which is a more generalized form of distribution than the Gaussian and Cauchy distributions.[1] Algorithms employing Lévy or Cauchy distributions, which both have a long fat tail, are more capable of escaping from local optima than the Gaussian counterpart, as suggested in several studies [22]–[24].

Without the velocity term, the Gaussian-based PSO (GPSO), as shown in (3), becomes similar to evolutionary programming (EP), where a Gaussian distribution is typically used for generating the next trial points in the search space [23]. One important difference though, is that in the Gaussian based PSO, the standard deviation σ is determined by the distance between $y_{i,d}$ and $\hat{y}_d$, whereas in a typical EP, σ needs to be supplied by the user, or be made self-adaptive [23], [24].

In another GPSO proposed by Secrest and Lamont [25], instead of sampling around the midpoint between $\mathbf{y}_i$ and $\hat{\mathbf{y}}$, a Gaussian distribution is used to sample around $\hat{\mathbf{y}}$ with some prespecified probability p, otherwise around $\mathbf{y}_i$. This proves to be beneficial, as particles can explore better in a much wider area, rather than just around the midpoint between $\mathbf{y}_i$ and $\hat{\mathbf{y}}$. However, since GPSO uses only a Gaussian distribution, its ability to explore the search space is rather limited, especially when the standard deviation σ becomes very small. In Section VI-A our experiments will demonstrate that using a combination of Cauchy and Gaussian distributions is superior to using only a Gaussian distribution for sampling, as also explained in the next section.

2) *lbest PSO:* One of the earliest PSO algorithms proposed was in fact an *lbest* PSO based on the ring topology [26]. A later study on PSO using a variety of neighbourhood topologies [27] showed that the *lbest* PSO based on the ring topology provided a slower convergence speed than the more widely used *gbest* PSO model. It is this slow convergence property that allows the *lbest* PSO to outperform the *gbest* model on a wide range of multimodal functions, though on unimodal functions, the *gbest* model is still likely to be the winner [28].

B. Cauchy and Gaussian-Based PSO

To solve large-scale optimization problems, an optimization algorithm needs to maintain its ability to explore effectively as well as to converge. Drawn from the findings of the aforementioned PSO variants, we propose a PSO model that employs both Cauchy and Gaussian distributions for sampling, as well as an *lbest* ring topology. The update rule for each particle position is rewritten as follows:

$$x_{i,d}(t+1) = \begin{cases} y_{i,d}(t) + \mathcal{C}(1)|y_{i,d}(t) - \hat{y}'_{i,d}(t)|, & \text{if } rand \le p \\ \hat{y}'_{i,d}(t) + \mathcal{N}(0,1)|y_{i,d}(t) - \hat{y}'_{i,d}(t)|, & \text{otherwise} \end{cases} \quad (4)$$

where $\mathcal{C}(1)$ denotes a number that is generated following a Cauchy distribution, and in this case we also need to set an "effective standard deviation" [22] for the Cauchy distribution, which is the same standard deviation value we would set for the equivalent Gaussian distribution, $|y_{i,d}(t) - \hat{y}'_{i,d}(t)|$; *rand* is a random number generated uniformly from $[0, 1]$; p is a user-specified probability value for Cauchy sampling to occur. Here, $\hat{\mathbf{y}}_i{}'$ denotes a local neighborhood best for the ith particle. Since an *lbest* ring topology is used for defining local neighborhood, $\hat{\mathbf{y}}_i{}'$ (i.e., the best-fit particle) is chosen among all three particles including the current ith particle and its immediate left and right neighbors (imagine that all particles are stored on a list that is indexed and wrapped-around). Since each particle may have a different $\hat{\mathbf{y}}_i{}'$, the population is likely to remain diverse for a longer period. The chance of prematurely converging to a single global best $\hat{\mathbf{y}}$, as in a GPSO (3), should be reduced. Note that p can be simply set to 0.5 so that half of the time Cauchy is used to sample around the ith particle's personal best $\mathbf{y}_i$ (more exploratory), while for the other half of the time Gaussian is used to sample around its neighborhood best $\hat{\mathbf{y}}_i{}'$ (less exploratory).

For the rest of the paper, this CGPSO is used as the subcomponent optimizer for each swarm in the context of a CC framework.

C. Dynamically Changing Group Size

When applying *random grouping* to CCEA, a group size s (i.e., the number of variables in each subcomponent) has to be chosen. Obviously, the choice of value for s will have a significant impact on the performance of CCPSO2. Furthermore, it might be desirable to vary s during a run. For example, it is possible to start with a smaller s value and then gradually increase this value over the course of the algorithm, in order to encourage convergence to a final global solution and to consider more potential variable interactions.

This issue was studied in a recently developed multilevel cooperative coevolution (MLCC) [19], where a scheme was proposed to probabilistically choose a group size from a set of potential group sizes. At the end of each coevolutionary cycle, a performance record list is used to update the probability values so that the group sizes associated with higher performances are rewarded with higher probability values accordingly. As a result, these more "successful" group sizes are more likely to be used again in future cycles.

This paper adopts an even simpler approach. At each iteration, we record the fitness value of the global best $\hat{\mathbf{y}}$ before and

[1]The shape of the Lévy distribution can be controlled by a parameter α. For $\alpha = 2$ it is equivalent to the Gaussian distribution, whereas for $\alpha = 1$ it is equivalent to the Cauchy distribution [22].

Algorithm 2: Pseudocode of CCPSO2

Create and initialize K swarms, each with s dimensions (where s is randomly chosen from a set $\mathbf{S}$, and $n = K * s$ always being true); the jth swarm is denoted as P_j, $j \in [1..K]$;
repeat
  if $f(\hat{\mathbf{y}})$ *has not improved* **then** randomly choose s from $\mathbf{S}$ and let $K = n/s$;
  Randomly permutate all n dimension indices;
  Construct K swarms, each with s dimensions;
  for *each swarm* $j \in [1..K]$ **do**
    for *each particle* $i \in [1..swarmSize]$ **do**
      if $f(\mathbf{b}(j, P_j.\mathbf{x}_i)) < f(\mathbf{b}(j, P_j.\mathbf{y}_i))$ **then**
      $P_j.\mathbf{y}_i \leftarrow P_j.\mathbf{x}_i$;
      if $f(\mathbf{b}(j, P_j.\mathbf{y}_i)) < f(\mathbf{b}(j, P_j.\hat{\mathbf{y}}))$ **then**
      $P_j.\hat{\mathbf{y}} \leftarrow P_j.\mathbf{y}_i$;
    end
    for *each particle* $i \in [1..swarmSize]$ **do**
      $P_j.\hat{\mathbf{y}}'_i \leftarrow \text{localBest}(P_j.\mathbf{y}_{i-1}, P_j.\mathbf{y}_i, P_j.\mathbf{y}_{i+1})$;
    end
    if $f(\mathbf{b}(j, P_j.\hat{\mathbf{y}})) < f(\hat{\mathbf{y}})$ **then** jth part of $\hat{\mathbf{y}}$ is replaced by $P_j.\hat{\mathbf{y}}$;
  end
  for *each swarm* $j \in [1..K]$ **do**
    for *each particle* $i \in [1..swarmSize]$ **do**
      Perform position update for the ith particle in swarm P_j using (4);
    end
  end
until *termination criterion is met*;

after a coevolutionary cycle,[2] and if there is no improvement of the fitness value, a new s is chosen uniformly at random from a set $\mathbf{S}$, otherwise, the value of s remains unchanged. Here $\mathbf{S}$ contains several possible s values ranging from small to large, e.g., $\mathbf{S} = \{2, 5, 50, 100, 200\}$. The idea is to continue to use the same s value as long as it works well, but if it does not, then a different value for s should be chosen. This simple scheme is demonstrated to work well experimentally (see Section VI-B). Although the set $\mathbf{S}$ still needs to be supplied, specification for the parameter s is no longer required. s values in $\mathbf{S}$ is applicable to any function of dimensions up to the largest value in $\mathbf{S}$.

One important difference between CCPSO2 and MLCC [19] is that the frequency of applying the random grouping method in CCPSO2 is likely to be much greater than that of MLCC. In CCPSO2 only one evolutionary step is executed over each subpopulation in a cycle, whereas in MLCC, it is common that several evolutionary steps are applied to each subpopulation. Given a fixed number of evaluations, random grouping is likely to be applied more frequently in CCPSO2 than MLCC. A higher frequency of applying random grouping should provide more benefit on nonseparable high dimensional problems, since it is more likely that interacting variables be captured in the same subcomponent of a CCEA [14].

[2]In CCPSO2 a coevolutionary cycle is one round-robin pass of coevolution of all subpopulations. Hence a cycle is equivalent to an iteration here.

D. CCPSO2

1) *Basic Algorithm:* Algorithm 2 summarizes the proposed CCPSO2 employing *random grouping* with dynamically changing group size s, as well as the Cauchy and Gaussian update rule (as given in (4) for a ring topology-based *lbest* PSO. Two nested loops are used to iterate through each swarm and each particle in that swarm. In the first nested loop, for the ith particle in the jth swarm, its personal best $P_j.\mathbf{x}_i$ is first checked for update, and similarly for the jth swarm best $P_j.\hat{\mathbf{y}}$. The function $localBest(.)$ returns $P_j.\hat{\mathbf{y}}_i'$, which is the best-fit particle in the local neighborhood of the ith particle, of the jth swarm. The jth swarm best $P_j.\hat{\mathbf{y}}$ is used to update the context vector $\hat{\mathbf{y}}$ if it is better. In the second nested loop, each particle's personal best, neighborhood best, and its corresponding swarm best are used to calculate the particle's next position using (4).

2) *Updating Personal Bests:* In order to update personal bests effectively in a coherent fashion over iterations, a new scheme is proposed here. Two matrices $\mathbf{X}$ and $\mathbf{Y}$ are used to store all information of particles' current positions and personal bests in all K swarms. For example, the ith row of $\mathbf{X}$ is a n-dimensional vector concatenating all current position vectors $\mathbf{x}_i$ from all K swarms (note that sw denotes $swarmSize$) as follows:

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{sw,1} & x_{sw,2} & \cdots & x_{sw,n} \end{bmatrix} \tag{5}$$

and

$$\mathbf{Y} = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,n} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{sw,1} & y_{sw,2} & \cdots & y_{sw,n} \end{bmatrix}. \tag{6}$$

When random grouping is applied, the indices of all columns are randomly permutated. These permutated indices are then used to construct K swarms over n dimensions, from $\mathbf{X}$ and $\mathbf{Y}$. This is achieved by simply taking out every s columns (or dimensions) of $\mathbf{X}$ and $\mathbf{Y}$ to form a new swarm. In other words, in each swarm a particle's position and personal best vectors are constructed according to the new permutated dimension indices. As a result of random grouping, the personal best of each newly formed particle in a swarm needs to be re-evaluated in order to obtain its correct "coevolving" fitness. This is a key difference from the conventional personal best update. All the experiments in later sections take these evaluations into account.

Similarly, $\hat{\mathbf{y}}$ should also be reconstructed according to the permutated dimension indices. This way, when forming new swarms based on the new permutated indices, the better position values found so far in certain dimensions can be used meaningfully to guide the search in future iterations.

Note that the original dimension indices are recorded so that when evaluating the ith particle of the jth swarm via function

$\mathbf{b}(j, P_j.\mathbf{x}_i)$ (which returns an n-dimensional vector consisting of $\hat{\mathbf{y}}$ with its jth component replaced by $P_j.\mathbf{x}_i$), the order of the original dimension indices of this n-dimensional vector can be always restored before invoking an evaluation function.

After evaluations of all particles in the jth swarm, if the swarm best $P_j.\hat{\mathbf{y}}$ is found to be better than the context vector $\hat{\mathbf{y}}$, then the corresponding part of $\hat{\mathbf{y}}$ gets replaced by $P_j.\hat{\mathbf{y}}$. This will ensure good information is recorded in $\hat{\mathbf{y}}$, and utilized in future iterations.

At the end of an iteration, after applying (4), the new current and personal best vectors are saved back to $\mathbf{X}$ and $\mathbf{Y}$. This will ensure that the next iteration can use the updated $\mathbf{X}$ and $\mathbf{Y}$ to start the process of applying random grouping again.

V. Experimental Studies

This section first describes test functions and performance measurements that are adopted, and then describes sep-CMA-ES, a state-of-the-art evolutionary strategy algorithm used in our comparative studies.

A. Experimental Setup

We adopt the seven benchmark test functions proposed for the CEC'08 special session on large-scale global optimization (LSGO) [5]. For each of these functions the global optimum is shifted with a different value in each dimension. While f_1 (*ShiftedSphere*), f_4 (*ShiftedRastrigin*), and f_6 (*ShiftedAckley*) are separable functions, f_2 (*SchwefelProblem*), f_3 (*ShiftedRosenbrock*), f_5 (*ShiftedGriewank*), and f_7 (*FastFractal*) are nonseparable, presenting a greater challenge to any algorithm that is prone to variable interactions. Note that f_5 (*ShiftedGriewank*) becomes easier to optimize as the number of dimensions increases, because the product component of f_5 becomes increasingly insignificant [29], making it more like a separable function (as variable interactions are almost negligible). In addition to the above seven CEC'08 functions, we also include four more functions, f_{3r}, f_{4r}, f_{5r}, and f_{6r}, which are rotated versions of f_3 f_4, f_5, and f_6, respectively. The effect of rotation introduces further variable interactions, making them nonseparable.[3] Rotations are performed in the decision space, on each plane using a random uniform rotation matrix [30], [31]. A new random uniform rotation matrix is generated for each individual run for the purpose of an unbiased assessment. By using these benchmark test functions, and the proposed set of evaluation criteria, we were able to compare CCPSO2 with other existing EAs.

Experiments were conducted on the above 11 test functions of 100, 500, and 1000 dimensions. The same performance measurements used for CEC'08 special session on LSGO were adopted [5]. For each test function, the averaged results of 25 independent runs were recorded. For each run, Max_FES (i.e., the maximum number of fitness evaluations) was set to 5000*n (where n is the number of dimensions). A two-tailed t-test was conducted with a null hypothesis stating that there is no difference between two algorithms in comparison. The null hypothesis was rejected if the p-value was smaller than the significance level $\alpha = 0.05$.

[3]Note that f_3 and f_5 are already nonseparable.

The population size for each swarm that participates in coevolution was set to 30. For random grouping of the variables, we used $\mathbf{S} = \{2, 5, 10, 50, 100\}$ for 100-D functions, and $\mathbf{S} = \{2, 5, 10, 50, 100, 250\}$ for 500-D and 1000-D functions, where $\mathbf{S}$ includes a range of possible group size values that can be dynamically chosen. A few further experiments on f_1, f_3, and f_7 of 2000 dimensions were also carried out, using the same setup as mentioned above.

The CGPSO described in Section IV-B was used in all experiments of CCPSO2, with p in (4) simply set to 0.5.

B. sep-CMA-ES

The CMA-ES algorithm, which makes use of adaptive mutation parameters through computing a covariance matrix and hence correlated step sizes in all dimensions, has been shown to be an efficient optimization method [32], [33]. However, most of the published results of CMA-ES were on functions of up to 100 dimensions [33], [34]. One major drawback of CMA-ES is its cost in calculating the covariance matrix, which has a complexity of $O(n^2)$. As dimensionality increases, this cost rapidly rises. Furthermore, sampling using a multivariate normal distribution and factorization of the covariance matrix also become increasingly expensive. In a recent effort to alleviate this problem [35], a simple modification to CMA-ES was introduced where only the calculation of the diagonal elements of the covariance matrix $\mathbf{C}$ is required. Thus, the complexity of updating $\mathbf{C}$ becomes linear with n. Since only $\mathbf{C}$ diagonal is utilized, the sampling becomes independent in each dimension, making this CMA-ES variant (so called sep-CMA-ES) no longer rotationally invariant. The tradeoff here is the much reduced cost as opposed to the deficiency in handling nonseparable problems. It was shown in [35] that sep-CMA-ES performs not only faster, but also surprisingly well on several separable and nonseparable functions of up to 1000 dimensions, outperforming the original CMA-ES.

In this paper, sep-CMA-ES [35], [36] was chosen to compare against the proposed CCPSO2. In our experiments, for sep-CMA-ES, each individual in the initial population is generated uniformly within the variable bounds $[A, B]^n$. The initial step-size σ is set to $(B - A)/2$. The population size was determined by n such that $\lambda = 4 + \lfloor 3\ln(n) \rfloor$, $\mu = \lfloor \frac{\lambda}{2} \rfloor$, where n is the dimensions of the problem, λ is the number of offspring, and μ is the number of parents, as suggested in [35]. A number of stopping criteria were used. For example, sep-CMA-ES stops if the range of the best objective function values of the last $10 + \lceil 30n/\lambda \rceil$ iterations is zero or below a small value specified, or if the standard deviation of the normal distribution is smaller than a specified threshold value in all coordinates, or if the maximal allowed number of evaluations is reached.

VI. Results and Analysis

This section consists of five parts. In the first part, we compare several PSO variants using different update rules, and demonstrate why CGPSO is the best subcomponent optimizer for CCPSO2. In the second part, we examine the effects of CCPSO2 using random grouping with changing group sizes.

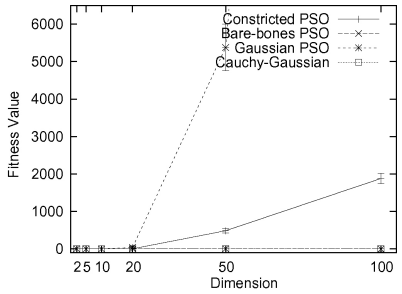

Fig. 2. Averaged best fitness values of four PSO variants (using a ring topology) on f_1 of 2, 5, 10, 20, 50, and 100 dimensions.

The results of CCPSO2 on 500-D functions are compared with those using a prespecified fixed group size. In the third part, we compare the results of both CCPSO2 and sep-CMA-ES on functions of 100, 500, and 1000 dimensions. We then compare CCPSO2 with two existing PSO algorithms and a cooperative coevolving DE on functions of 1000 dimensions, for which the results were obtained from the CEC 2008 competition on LSGO [37]. In the final part we present results on functions of 2000 dimensions in order to further challenge CCPSO2's ability to scale to even higher dimensions.

A. Why Cauchy and Gaussian PSO?

In Section IV-B, we provided the rationale of choosing the CGPSO as the subcomponent optimizer for optimizing each swarm under the framework of CC. In this section, we present the empirical evidence to support this choice. For a function of 1000 dimensions, s may range from several dimensions up to several hundreds. In this paper, we used four PSO variants using the ring topology, including constricted PSO (CPSO) [i.e., using (1) and (2)], Bare-bones PSO (3), GPSO [25], and CGPSO (4). We chose f_1 to f_7 of 2, 5, 10, 20, 50, and 100 dimensions. Using a small population of 30, each algorithm was run 50 times (with each run allowing 300 000 FES), and the mean and standard error of the best fitness were recorded. As shown in Figs. 2 and 3, for f_1 to f_7 up to 50 dimensions, CGPSO is the overall best performer. However, when the dimension was increased to 100, CGPSO's lead is no longer obvious. CGPSO outperformed CPSO on f_1, f_4, and f_5, but lost to CPSO on f_2, f_3, and f_6. Both CGPSO and CPSO performed similarly on f_7. Note that the global optimum of f_7 is usually unknown (unlike other functions) and may change depending on the dimensionality. Results of the other two variants, Bare-bones PSO and GPSO were not competitive. In particular, GPSO tended to prematurely converge very quickly on several functions. Comparing GPSO and CGPSO, it is noticeable that using a combination of Cauchy and Gaussian distributions is far more effective than using only a Gaussian distribution for sampling. Overall, CGPSO appears to be the most consistent and better performer over other variants. Typically for a function of 500 dimensions, a subcomponent size (i.e., group size s) ranging from 2 to 100 dimensions is most likely to be adaptively chosen (see next section) during the CC process. Hence, we selected CGPSO as the subcomponent optimizer for each swarm of CCPSO2.

B. Effects of Dynamically Changing Group Size

Table I compares the results of CCPSO2 using dynamically changing group size s and those using a prespecified fixed group size. When different fixed group sizes are used, it is noticeable that performance fluctuates a lot, depending heavily on the given s value. On the contrary, CCPSO2, using a dynamically changing s value, consistently gave a better performance than the other variants. The closest rival to CCPSO2 is the $s = 50$ variant, which only outperformed CCPSO2 on f_2. However, in most real-world problems, we do not have any prior knowledge about the optimal s value.

Among the CCPSO2 variants using a fixed s value, it is interesting to note that it is not necessarily always better to use a small fixed s even for separable functions such as f_1. The $s = 100$ variant in fact is much better than the $s = 5$ variant. The better performance of CCPSO2 using a dynamically changing s may be attributed to the diverse range of s values that CCPSO2 can utilize during a run. CCPSO2 can use smaller s values to evolve small subcomponent-based intermediate solutions first and then uses larger s values to evolve the combined intermediate solutions to achieve even fitter overall solutions.

Fig. 4 shows typical CCPSO2 runs with a dynamically changing group size on the CEC'08 test functions. It is noticeable that for separable functions f_1, f_4, f_6, and f_5 (which is considered as a more separable function for 500-D), CCPSO2 tended to choose a mixture of small and large group sizes at different stages of a run. For nonseparable functions such as f_2, f_3, and f_7, CCPSO2 tended to favor a small group size throughout the run. Since the group size remains unchanged when there is further performance improvement, this suggests CCPSO2 was still able to improve the performance though very slowly.

C. Comparing CCPSO2 with sep-CMA-ES

Table II shows the results of the 11 test functions of 100-D, 500-D, and 1000-D. The result of sep-CMA-ES scaled very well from 100-D to 1000-D on f_1, f_3, and f_{4r}, outperforming CCPSO2 on all these cases. On f_5 and f_{5r}, the performance of sep-CMA-ES is better on 100-D, but not statistically different on 500-D and 1000-D. Figs. 5–8 show the convergence plots where it can be noted the sep-CMA-ES achieved very fast convergence on f_1, f_3, f_5, and f_{5r}.

In comparison, CCPSO2 clearly outperformed sep-CMA-ES on f_2, f_4, f_6, and f_7. On f_{6r}, CCPSO2 performed better on 100-D and 500-D, but its performance is not statistically different from sep-CMA-ES on 1000-D. There is also no statistical difference on f_{3r}. It can be also noted that rotation of f_4 into f_{4r} degraded the performance of CCPSO2, but had little effect on sep-CMA-ES.

On f_2, f_4, f_6, and f_{6r}, sep-CMA-ES prematurely converged as soon as a run started.[4] sep-CMA-ES also had a poor performance compared with CCPSO2 on f_7 *FastFractal* function, which has a more complex and irregular fitness landscape, and highly multimodal. The standard CMA-ES (which uses

[4]The run may be terminated if no further improvement after several iterations as described in Section V-B.

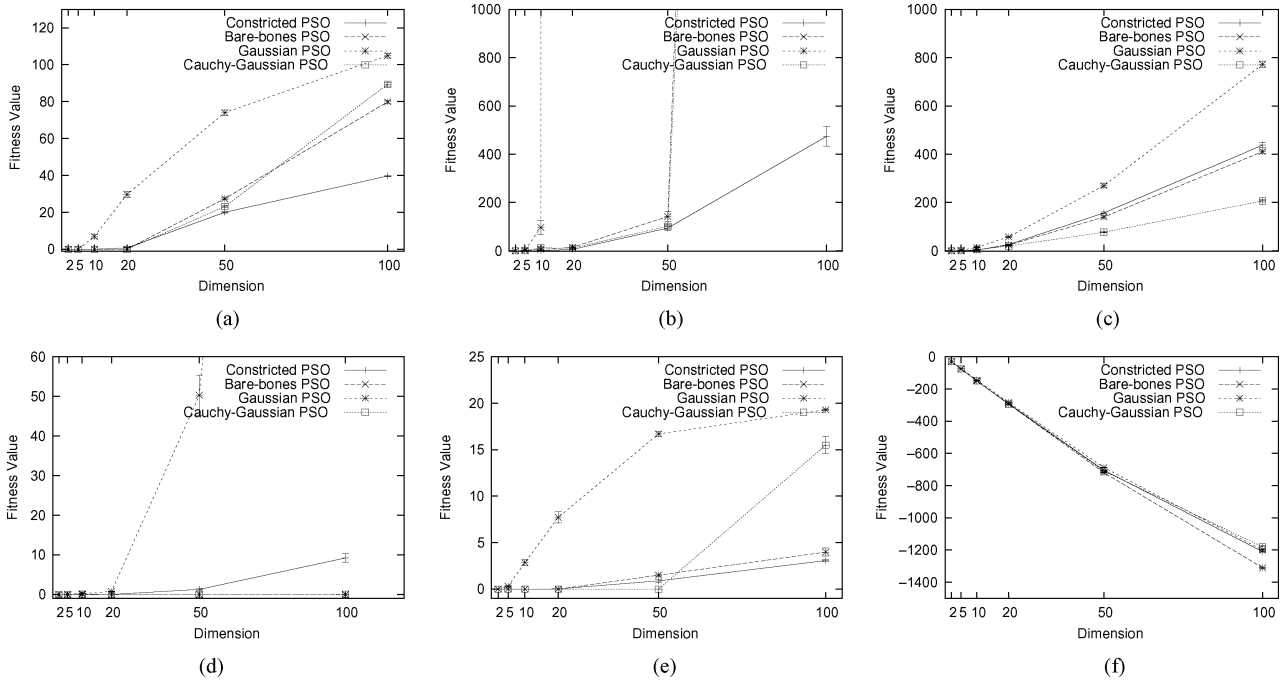

Fig. 3. Averaged best fitness values (with one standard error) of four PSO variants using a ring topology on f_2 to f_7 of 2, 5, 10, 20, 50, and 100 dimensions. (a) f_2 SchwefelProblem. (b) f_3 ShiftedRosenbrock. (c) f_4 ShiftedRastrigin. (d) f_5 ShiftedGriewank. (e) f_6 ShiftedAckley. (f) f_7 FastFractal.

TABLE I

RESULTS ON 500-D FUNCTIONS OF CCPSO2 USING RANDOM GROUPING WITH A CHANGING GROUP SIZE AND THOSE CCPSO2 VARIANTS USING A PRESPECIFIED FIXED GROUP SIZE

500-D	$s \in S$	$s = 5$	$s = 10$	$s = 50$	$s = 100$	p-value
f_1	**3.07E-13 (7.34E-14)**	4.54E+05 (1.06E+06)	7.57E-13 (1.03E-13)	3.52E-13 (4.02E-14)	3.87E-13 (2.84E-14)	0.011
f_2	6.37E+01 (2.89E+01)	9.47E+01 (6.21E+01)	8.27E+01 (7.64E+01)	**1.50E+01 (3.44E+01)**	4.83E+01 (3.30E+00)	0.000
f_3	7.51E+02 (1.61E+02)	3.78E+11 (1.04E+12)	1.18E+11 (5.88E+11)	8.84E+02 (9.34E+01)	6.81E+02 (1.43E+02)	0.001
f_4	**1.59E-01 (3.72E-01)**	9.55E+02 (3.30E+03)	1.43E+03 (3.95E+03)	2.77E+02 (5.24E+01)	5.87E+02 (5.72E+01)	0.000
f_5	9.85E-04 (3.41E-03)	3.83E+03 (8.97E+03)	1.94E+03 (6.72E+03)	2.96E-04 (1.48E-03)	9.85E-04 (3.69E-03)	0.361
f_6	**5.50E-13 (8.79E-14)**	2.57E+00 (7.10E+00)	1.72E+00 (5.94E+00)	6.71E-13 (2.46E-14)	2.35E-09 5.08E-09)	0.000
f_7	**-7.22E+03 (5.37E+01)**	-5.04E+03 (1.94E+03)	-7.07E+03 (7.79E+02)	-6.35E+03 (1.05E+02)	-5.68E+03 (3.03E+02)	0.000

Note that $S = \{2, 5, 10, 50, 100, 250\}$. A two-tailed t-test was conducted between CCPSO2 using a changing group size from S and one using $s = 50$. If two results are statistically different, the better one is highlighted in bold.

the full covariance matrix) was also tested on f_2, f_4, and f_6, but CMA-ES suffered from the same problem of premature convergence.

The offspring population size λ of a sep-CMA-ES run was determined by $\lambda = 4 + \lfloor 3\ln(n) \rfloor$. For n equals to 100, 500, and 1000, λ is 17, 22, and 24, respectively. Further tests revealed that these population sizes were too small for sep-CMA-ES (or CMA-ES) to perform well on f_2, f_4, f_6, and f_7, though smaller population sizes did not pose any trouble on f_1, f_3, and f_5. For sep-CMA-ES to perform well, it requires a much larger population size in order to sample effectively around the mean of the selected fit individuals. This suggests that the formula proposed for determining the population size $\lambda = 4 + \lfloor 3\ln(n) \rfloor$ is not suitable for multimodal functions of higher dimensions ($n > 100$). This observation is also supported in [38], where a study was carried out specifically on CMA-ES over multimodal functions. It shows that on the Rastrigin function of only 80 dimensions and Schwefel of 20 dimensions, CMA-ES requires a λ greater than 1000 in order to maintain a success rate of 95% (Figs. 2 and 3 of [38]). In [35], a different formula for determining population size was also studied: $\lambda = 2n$. Nevertheless this means that the population will be unrealistically large for functions of 1000 dimensions for instance. Fig. 9(a) shows the results of sep-CMA-ES on f_4 (*ShiftedRastrigin*) of 500 dimensions, over population sizes ranging from 100 to 1500. It can be seen that as the population size increases, the performance also improves, however, even with a population size of 1500, the best fitness achieved was 14.725, which is still worse than CCPSO2. Fig. 9(b) shows that on the more complex f_7 (*FastFractal*), the best result achieved was -7057.21 (when a population size of 400 was used), which is still worse than the best result obtained by CCPSO2 (i.e., -7.23E+03 as shown in Table II). Clearly, sep-CMA-ES's performance is sensitive to the population size parameter. And even if the optimal population size is chosen, CCPSO2 still outperforms sep-CMA-ES on these multimodal functions. More importantly, the performance of CCPSO2 does not depend on a large

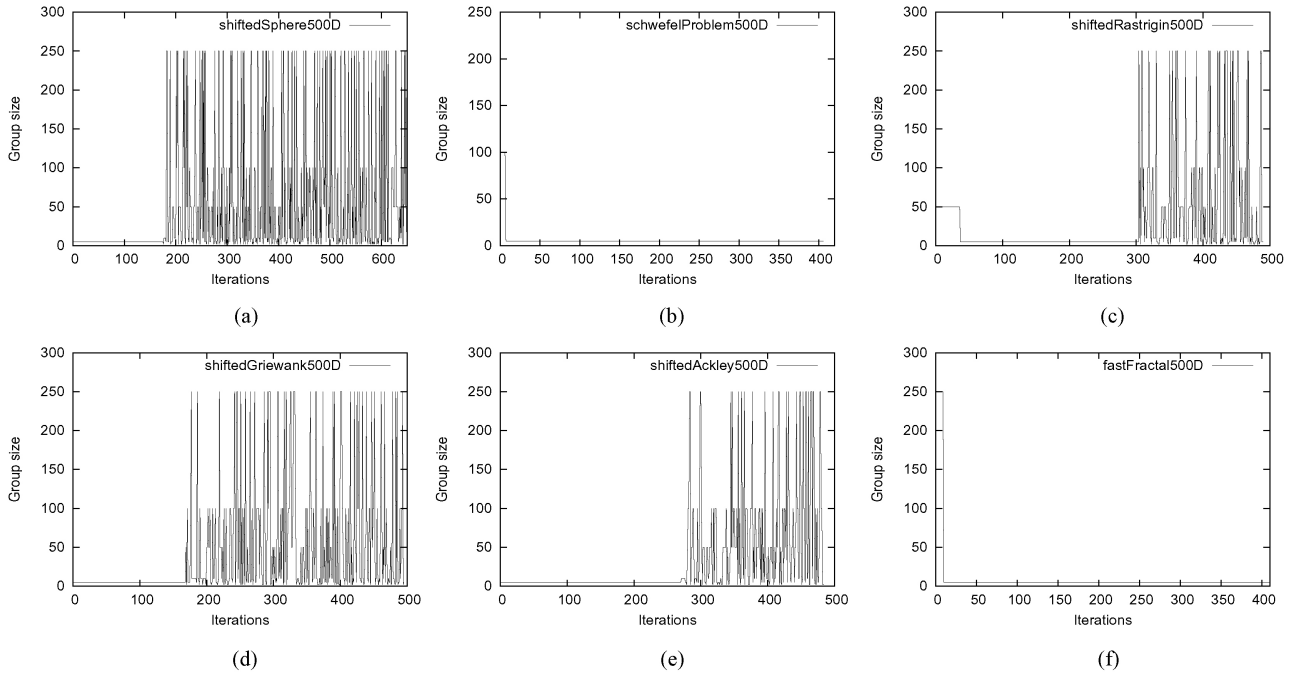

Fig. 4. Changing group size during a CCPSO2 run on f_1 to f_7 of 500 dimensions. (The result of f_3 ShiftedRosenbrock is not shown as it has a unchanged group size of 5 over the run.) (a) f_1 ShiftedSphere. (b) f_2 SchwefelProblem. (c) f_4 ShiftedRastrigin. (d) f_5 ShiftedGriewank. (e) f_6 ShiftedAckley. (f) f_7 FastFractal.

TABLE II
RESULTS OF CCPSO2 AND SEP-CMA-ES ON TEST FUNCTIONS OF 100-D, 500-D, AND 1000-D

Test function	Dimensions	CCPSO2	sep-CMA-ES	p-value
f_1	100-D	7.73E-14 (3.23E-14)	**9.02E-15 (5.53E-15)**	0.000
	500-D	3.00E-13 (7.96E-14)	**2.25E-14 (6.10E-15)**	0.000
	1000-D	5.18E-13 (9.61E-14)	**7.81E-15 (1.52E-15)**	0.000
f_2	100-D	**6.08E+00 (7.83E+00)**	2.31E+01 (1.39E+01)	0.000
	500-D	**5.79E+01 (4.21E+01)**	2.12E+02 (1.74E+01)	0.000
	1000-D	**7.82E+01 (4.25E+01)**	3.65E+02 (9.02E+00)	0.000
f_3	100-D	4.23E+02 (8.65E+02)	**4.31E+00 (1.26E+01)**	0.023
	500-D	7.24E+02 (1.54E+02)	**2.93E+02 (3.59E+01)**	0.000
	1000-D	1.33E+03 (2.63E+02)	**9.10E+02 (4.54E+01)**	0.000
f_4	100-D	**3.98E-02 (1.99E-01)**	2.78E+02 (3.43E+01)	0.000
	500-D	**3.98E-02 (1.99E-01)**	2.18E+03 (1.51E+02)	0.000
	1000-D	**1.99E-01 (4.06E-01)**	5.31E+03 (2.48E+02)	0.000
f_5	100-D	3.45E-03 (4.88E-03)	**2.96E-04 (1.48E-03)**	0.004
	500-D	1.18E-03 (4.61E-03)	7.88E-04 (2.82E-03)	0.719
	1000-D	1.18E-03 (3.27E-03)	3.94E-04 (1.97E-03)	0.310
f_6	100-D	**1.44E-13 (3.06E-14)**	2.12E+01 (4.02E-01)	0.000
	500-D	**5.34E-13 (8.61E-14)**	2.15E+01 (3.10E-01)	0.000
	1000-D	**1.02E-12 (1.68E-13)**	2.15E+01 (3.19E-01)	0.000
f_7	100-D	**-1.50E+03 (1.04E+01)**	-1.39E+03 (2.64E+01)	0.000
	500-D	**-7.23E+03 (4.16E+01)**	-6.37E+03 (7.59E+01)	0.000
	1000-D	**-1.43E+04 (8.27E+01)**	-1.25E+04 (9.36E+01)	0.000
f_{3r}	100-D	7.56E+02 (1.67E+03)	4.11E+02 (1.20E+03)	0.406
	500-D	6.49E+02 (6.05E+02)	4.96E+02 (1.82E+02)	0.236
	1000-D	2.64E+03 (3.30E+03)	3.94E+03 (1.34E+04)	0.642
f_{4r}	100-D	8.29E+02 (2.22E+02)	**2.71E+02 (3.85E+01)**	0.000
	500-D	4.97E+03 (8.01E+02)	**2.18E+03 (1.22E+02)**	0.000
	1000-D	9.84E+03 (1.15E+03)	**5.54E+03 (2.31E+02)**	0.000
f_{5r}	100-D	2.66E-03 (4.60E-03)	**4.93E-04 (2.46E-03)**	0.045
	500-D	2.96E-04 (1.48E-03)	2.96E-04 (1.48E-03)	1.000
	1000-D	6.90E-04 (2.41E-03)	1.29E-12 (3.09E-13)	0.165
f_{6r}	100-D	**8.94E+00 (8.92E+00)**	2.14E+01 (2.42E-02)	0.000
	500-D	**1.97E+01 (3.94E-01)**	2.15E+01 (3.03E-01)	0.000
	1000-D	1.97E+01 (1.12E-01)	2.07E+01 (4.32E+00)	0.259

A two-tailed t-test was conducted between CCPSO2 and sep-CMA-ES. If two results are statistically different, the better one is highlighted in bold.

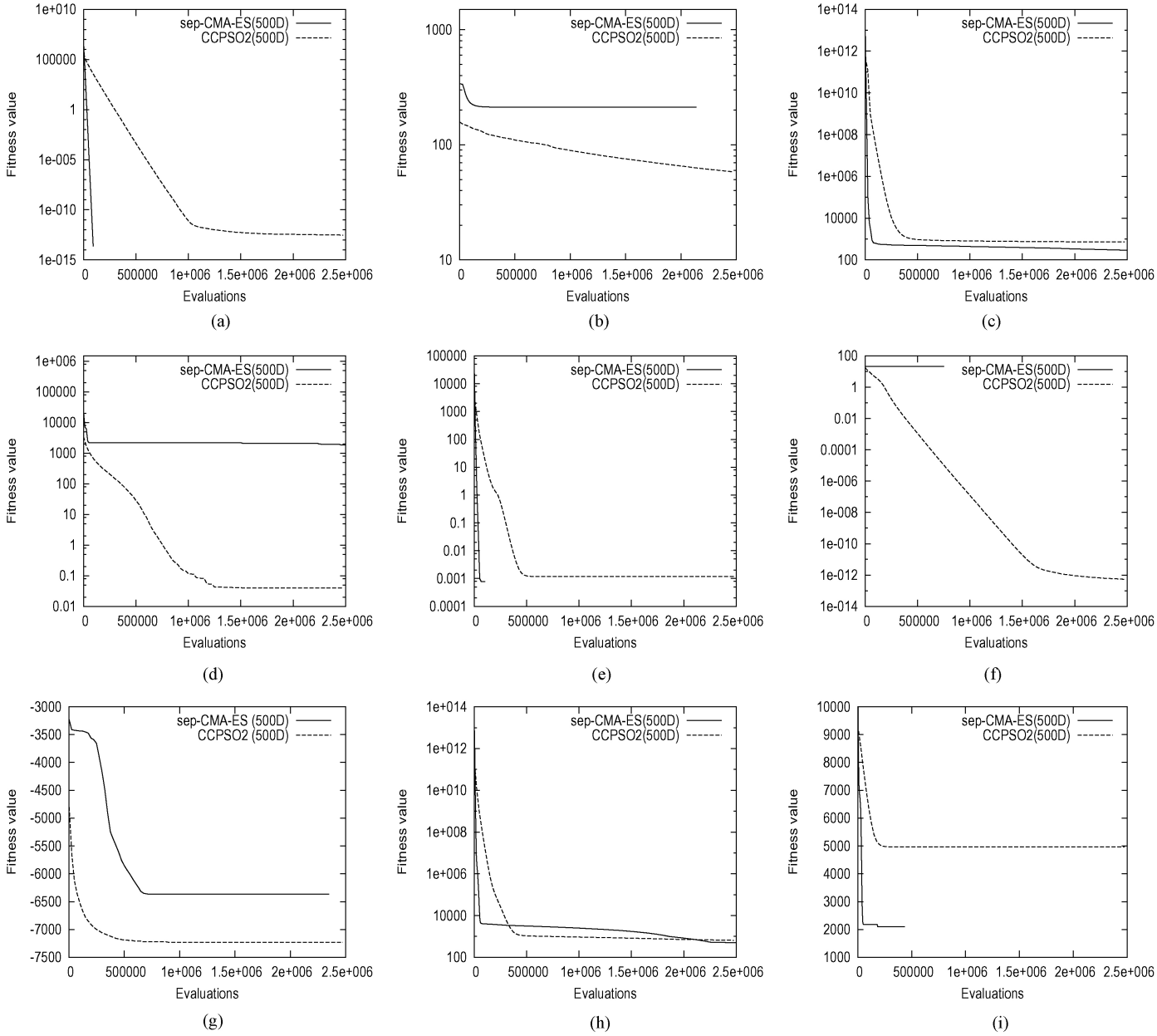

Fig. 5. Averaged best fitness values for sep-CMA-ES and CCPSO2 on functions of 500 dimensions. (a) f_1 ShiftedSphere. (b) f_2 SchwefelProblem. (c) f_3 ShiftedRosenbrock. (d) f_4 ShiftedRastrigin. (e) f_5 ShiftedGriewank. (f) f_6 ShiftedAckley. (g) f_7 ShiftedRastrigin. (h) f_{3r} ShiftedRotatedRosenbrock. (i) f_{4r} ShiftedRotatedRastrigin.

population size. For all experiments, a small swarm of 30 particles for a coevolving subcomponent was shown to be sufficient in producing reasonable performances.

Table II shows that overall, for the 11 test functions, CCPSO2 performed significantly better than sep-CMA-ES on five functions, while losing to sep-CMA-ES on three functions. For the remaining three test functions, there is no statistical difference between the two. Most importantly, CCPSO2 performed better than sep-CMA-ES on high-dimensional multimodal functions with a complex fitness landscape such as f_7, which arguably more closely resemble real-world problems. The CC framework allowing decomposing a high dimensional problem into small subcomponents is a key contributing factor that CCPSO2 scales well to very high-dimensional problems.

CCPSO2 tended to converge slower than sep-CMA-ES, but had more potential to further improve its performance in the later stage of a run. This may be attributed to the use of the Cauchy distribution in CCPSO2 to encourage more exploration. On the contrary, sep-CMA-ES converged extremely fast, however, it either converged very well or tended to become stagnant very quickly, especially on complex high dimensional multimodal functions.

D. Comparing CCPSO2 with Other Algorithms

Table III shows the results of CCPSO2 on the seven CEC'08 benchmark test functions of 1000 dimensions, in comparison with two recently proposed PSO variants, EPUS-PSO [39], DMS-PSO [40], and a CC DE algorithm MLCC [19]. The results of these algorithms were obtained under the same

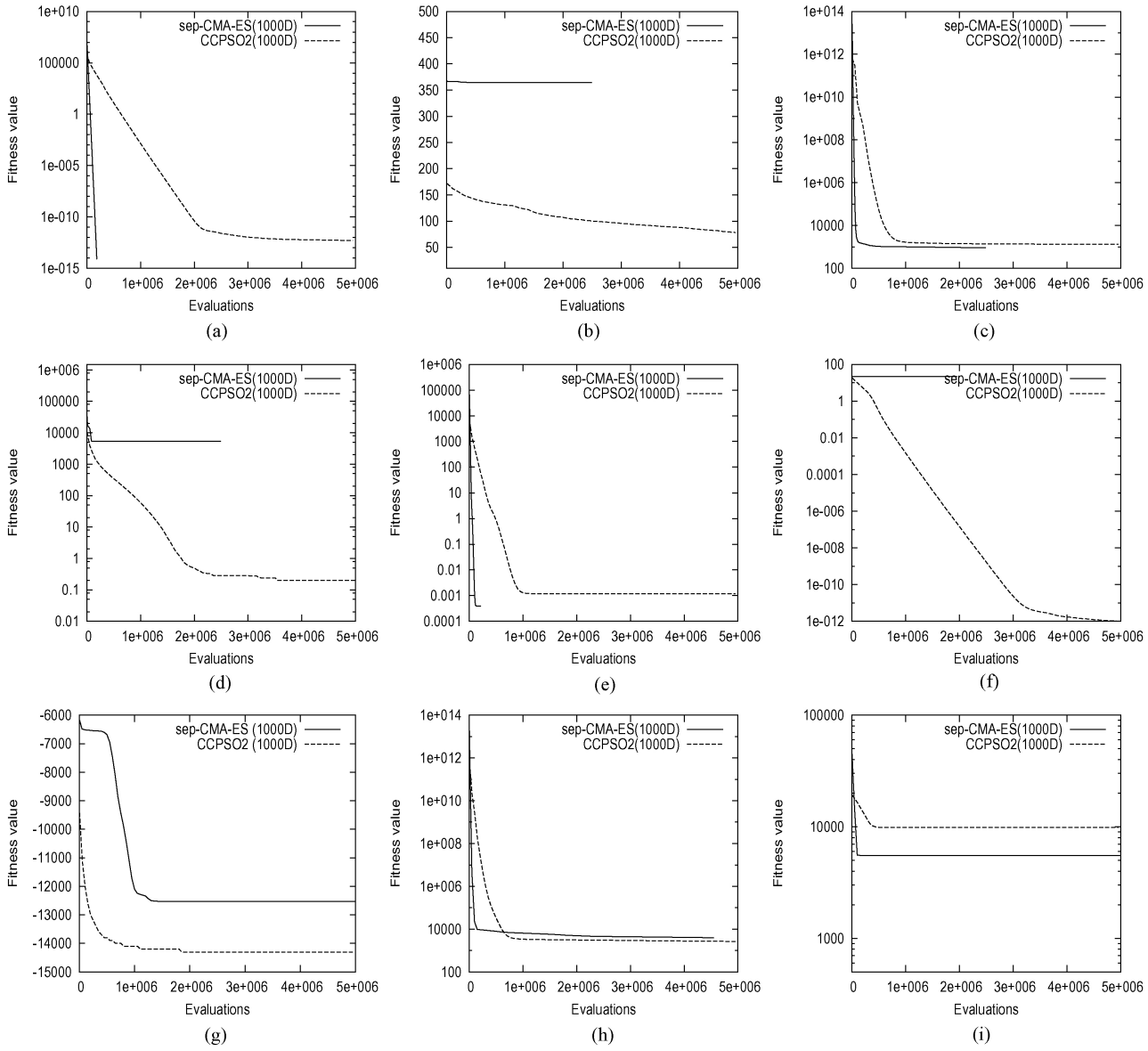

Fig. 6. Averaged best fitness values for sep-CMA-ES and CCPSO2 on functions of 1000 dimensions. (a) f_1 ShiftedSphere. (b) f_2 SchwefelProblem. (c) f_3 ShiftedRosenbrock. (d) f_4 ShiftedRastrigin. (e) f_5 ShiftedGriewank. (f) f_4 ShiftedRastrigin. (g) f_7 ShiftedRastrigin. (h) f_{3r} ShiftedRotatedRosenbrock. (i) f_{4r} ShiftedRotatedRastrigin.

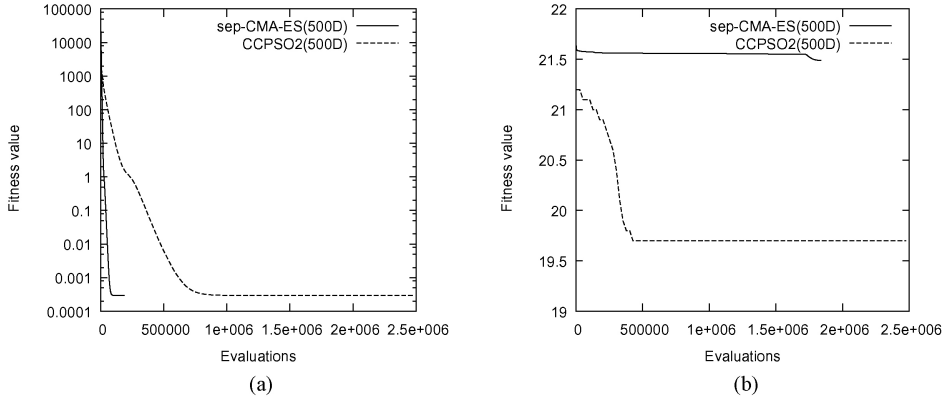

Fig. 7. Averaged best fitness values for sep-CMA-ES and CCPSO2 on f_{5r} and f_{6r} of 500 dimensions. (a) f_{5r}. (b) f_{6r}.

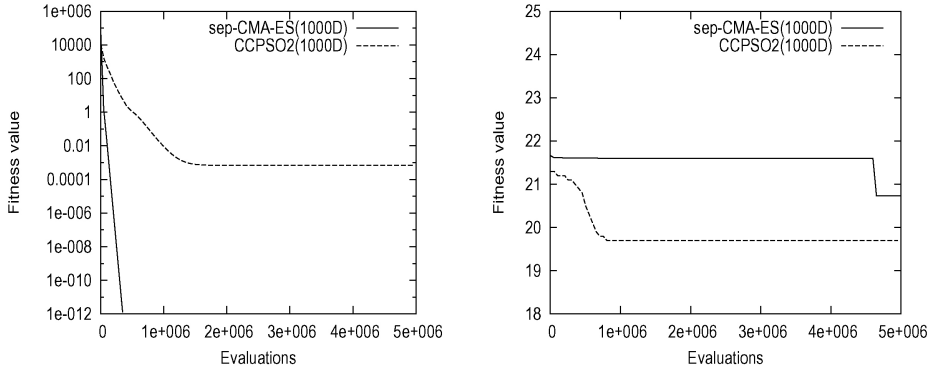

Fig. 8. Averaged best fitness values for sep-CMA-ES and CCPSO2 on f_{5r} and f_{6r} of 1000 dimensions. (a) f_{5r}. (b) f_{6r}.

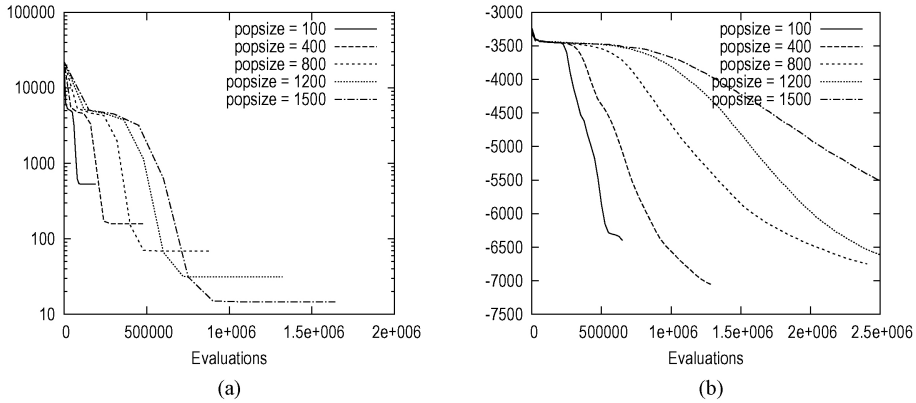

Fig. 9. Averaged best fitness values of sep-CMA-ES on f_4 and f_7 of 500 dimensions, over a range of population sizes. (a) f_4 ShiftedRastrigin. (b) f_7 FastFractal.

TABLE III
COMPARISON BETWEEN CCPSO2, EPUS-PSO, DMS-PSO, AND MLCC AND ON 1000-D FUNCTIONS

1000-D	CCPSO2	EPUS-PSO	DMS-PSO	MLCC	p-value
f_1	**5.18E-13 (9.61E-14)**	5.53E+02 (2.86E+01)	0.00E+00 (0.00E+00)	8.46E-13 (5.01E-14)	0.000
f_2	**7.82E+01 (4.25E+01)**	4.66E+01 (4.00E-01)	9.15E+01 (7.14E-01)	1.09E+02 (4.75E+00)	0.001
f_3	**1.33E+03 (2.63E+02)**	8.37E+05 (1.52E+05)	8.98E+09 (4.39E+08)	1.80E+03 (1.58E+02)	0.000
f_4	1.99E-01 (4.06E-01)	7.58E+03 (1.51E+02)	3.84E+03 (1.71E+02)	**1.37E-10 (3.37E-10)**	0.022
f_5	1.18E-03 (3.27E-03)	5.89E+00 (3.91E-01)	0.00E+00 (0.00E+00)	4.18E-13 (2.78E-14)	0.084
f_6	1.02E-12 (1.68E-13)	1.89E+01 (2.49E+00)	7.76E+00 (8.92E-02)	1.06E-12 (7.68E-14)	0.287
f_7	-1.43E+04 (8.27E+01)	-6.62E+03 (3.18E+01)	-7.51E+03 (1.64E+01)	**-1.47E+04 (1.52E+01)**	0.000

Results for EPUS-PSO, DMS-PSO, and MLCC quoted from [19], [39], and [40]. A two-tailed t-test was conducted between CCPSO2 and MLCC. If two results are statistically different, the better one is highlighted in bold.

criteria, as set out by the CEC'08 special session on LSGO [5]. A two-tailed t-test was only conducted between CCPSO2 and MLCC, since CCPSO2 clearly outperformed EPUS-PSO and DMS-PSO.

EPUS-PSO used an adaptive population sizing strategy to adjust the population size according to the search results [39]. DMS-PSO adopted a random regrouping strategy to introduce a dynamically changing neighborhood structure to each particle. Every now and then the population is regrouped into multiple small subpopulations according to the new neighborhood structures [40]. MLCC also adopts a CC approach (like CCPSO2) applying both random grouping and adaptive weighting [19]. Given a fixed number of evaluations, the applications of adaptive weighting is at the expense of random grouping, which proves to be less effective [14]. In addition, MLCC employs a self-adaptive mechanism to favor certain group size over others when invoking the random grouping scheme. All these three algorithms participated in the CEC 2008 competition on LSGO [37].

CCPSO2 outperformed EPUS-PSO on six out of the seven CEC'08 test functions, except f_2. CCPSO2 also outperformed DMS-PSO on five functions, except f_1 and f_5. DMS-PSO found the global optimum for f_1 and f_5, but its performance is exceptionally poor in comparison with CCPSO2 on the other five functions. CCPSO2 adopts a CC framework to decompose a high-dimensional problem, whereas DMS-PSO relies on the use of a large number of small sub-swarms (each has only three particles) to maintain its population diversity. DMS-PSO does not employ any dimensionality decomposition strategy. As a result, DMS-PSO uses much larger population

TABLE IV
RESULTS OF CCPSO2 AND SEP-CMA-ES ON f_1, f_3, AND f_7 OF 2000 DIMENSIONS

2000-D	CCPSO2	sep-CMA-ES	p-value
f_1	1.03E-12 (2.56E-13)	**6.03E-15 (6.14E-16)**	0.000
f_3	2.91E+03 (6.43E+02)	**1.73E+03 (7.77E+01)**	0.000
f_7	**-2.86E+04 (1.90E+02)**	-2.46E+04 (1.57E+02)	0.000

A two-tailed t-test was conducted between CCPSO2 and MLCC.

sizes to handle high-dimensional problems. For example, for 100, 500, and 1000-D functions, the population sizes were 90, 450, and 900, respectively. In contrast, CCPSO2 uses just a small population size of 30 for all dimensions. This suggests that DMS-PSO may only work well in some unimodal fitness landscapes, but has a very poor ability in handling complex multimodal fitness landscapes.

Comparing with MLCC, CCPSO2 is better on f_1, f_2, and f_3, but statistically not different on f_5 and f_6. MLCC is better on f_4 and f_7. MLCC uses a self-adaptive neighborhood search DE (SaNSDE) [12] as its core algorithm to coevolve its CC subcomponents. SaNSDE is able to maintain a better diversity of search step sizes and the population, thereby contributing to its better performance on the multimodal problems.

E. Comparisons on Functions of 2000 Dimensions

From the previous Section VI-C (Table II, Figs. 5 and 6), we noticed that on f_2, f_4, and f_6 of 100, 500, and 1000 dimensions, sep-CMA-ES always prematurely converged not long after the run started. sep-CMA-ES was actually outperformed substantially by CCPSO2 on these functions. It is reasonable to expect that sep-CMA-ES be outperformed by CCPSO2 on these functions of even higher dimensions such as 2000-D.

Table IV shows the results of CCPSO2 and sep-CMA-ES on the three other CEC'08 test functions: f_1, f_3, and f_7 of 2000 dimensions. These results were averaged over 25 runs, with each run consuming 1.0E+07 FES. To our knowledge, this paper is the first reporting results on these functions of 2000 dimensions. Overall, we can see that both CCPSO2 and sep-CMA-ES continued to scale well on f_1 and f_3 of 2000 dimensions. sep-CMA-ES performed better than CCPSO2 on f_1 (which is a separable function) and f_3, but was outperformed by CCPSO2 on the more complex multimodal f_7. Considering the tendency of sep-CMA-ES converging prematurely on multimodal functions f_2, f_4, and f_6, once again CCPSO2 shows its better search capability in handling complex high-dimensional multimodal functions.

VII. CONCLUSION

In this paper, we presented a new CC PSO, CCPSO2, for tackling large scale optimization problems. We demonstrated that the CC framework adopted is a powerful approach to improving PSO's ability in scaling up to high-dimensional optimization problems (of up to 2000 real-valued variables). Several new techniques have been incorporated into CCPSO2 to enhance its ability to handle high-dimensional problems: a novel PSO model (CGPSO) was developed to sample a particle's next point following a combination of Cauchy and Gaussian distributions; this CGPSO adopts an *lbest* ring topology for defining its local neighborhood. The algorithm employed the random grouping scheme with a dynamically changing group size. CGPSO showed improved search capability compared with existing PSO models. The effects of using a dynamically changing group size has shown that on most high-dimensional functions, a combination of small and reasonably large group sizes are advantageous.

Furthermore, CCPSO2 was compared with a state-of-the-art evolutionary algorithm sep-CMA-ES. Our results showed that with only a small population size, CCPSO2 was very competitive on high-dimensional functions having a more complex multimodal fitness landscape such as f_7 (*FastFractal*), though sep-CMA-ES performed slightly better on unimodal functions. Our findings also reveal that the performance of sep-CMA-ES could degrade rapidly on a multimodal fitness landscape, even if a very large population size was chosen. Further comparative studies on 1000-D functions suggest that CCPSO2's performance is comparable to a CC DE algorithm [19], and much better than two other PSO algorithms which were specifically designed for large scale optimization.

We also challenged CCPSO2 with functions of 2000 dimensions, and the results show that CCPSO2 continued to outperform sep-CMA-ES on the more complex multimodal function f_7 (*FastFractal*), while performing reasonably well on the unimodal functions.

In future, we are planning to examine more "intelligent" grouping strategies to better capture the interdependency among variables [41], rather than just random grouping. We are also interested in applying CCPSO2 to real-world problems such as shape optimization [42] to ascertain its true potential as a valuable optimization technique for large-scale optimization.

ACKNOWLEDGMENT

The authors would like to thank Z. Yang for insightful discussions on the working of DECC-G and M. N. Omidvar for valuable comments.

REFERENCES

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

[2] T. Sonoda, Y. Yamaguchi, T. Arima, M. Olhofer, B. Sendhoff, and H.-A. Schreiber, "Advanced high turning compressor airfoils for low Reynolds number condition, part I: Design and optimization," *J. Turbomach.*, vol. 126, no. 3, pp. 350–359, 2004.

[3] A. Vicini and D. Quagliarella, "Airfoil and wing design through hybrid optimization strategies," in *Proc. 16th Appl. Aerodynamics Conf.*, AIAA paper 98-2729. 1998, pp. 1–11.

[4] K. Foli, T. Okabe, M. Olhofer, Y. Jin, and B. Sendhoff, "Optimization of micro heat exchanger: CFD, analytical results and multiobjective evolutionary algorithms," *Int. J. Heat Mass Transfer*, vol. 49, nos. 5–6, pp. 1090–1099, 2006.

[5] K. Tang, X. Yao, P. Suganthan, C. MacNish, Y. Chen, C. Chen, and Z. Yang, "Benchmark functions for the CEC'2008 special session and competition on large scale global optimization," Nature Inspired Computat. Applicat. Lab., Univ. Sci. Technol. China, Hefei, China, Tech. Rep., 2007 [Online]. Available: http://nical.ustc.edu.cn/cec08ss.php

[6] K. Tang, X. Li, P. Suganthan, Z. Yang, and T. Weise, "Benchmark functions for the CEC'2010 special session and competition on large scale global optimization," Nature Inspired Computat. Applicat. Lab., Univ. Sci. Technol. China, Hefei, China, Tech. Rep., 2009 [Online]. Available: http://nical.ustc.edu.cn/cec10ss.php

[7] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proc. IEEE CEC*, vol. 2. Jun. 2004, pp. 1980–1987.
[8] F. van den Bergh and A. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.
[9] M. Potter and K. D. Jong, "A cooperative coevolutionary approach to function optimization," in *Proc. 3rd Conf. Parallel Problem Solving Nat.*, 1994, pp. 249–257.
[10] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Inform. Sci.*, vol. 178, no. 15, pp. 2986–2999, Aug. 2008.
[11] F. V. den Bergh, "An analysis of particle swarm optimizers," Ph.D. dissertation, Dept. Comput. Sci., Univ. Pretoria, Pretoria, South Africa, 2002.
[12] Z. Yang, K. Tang, and X. Yao, "Differential evolution for high-dimensional function optimization," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 3523–3530.
[13] X. Li and X. Yao, "Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms," in *Proc. IEEE CEC*, May 2009, pp. 1546–1553.
[14] M. Omidvar, X. Li, X. Yao, and Z. Yang, "Cooperative co-evolution for large scale optimization through more frequent random grouping," in *Proc. CEC*, Jul. 2010, pp. 1754–1761.
[15] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution," in *Proc. IEEE Congr. Evol. Comput.*, May 2001, pp. 1101–1108.
[16] D. Sofge, K. D. Jong, and A. Schultz, "A blended population approach to cooperative coevolution for decomposition of complex problems," in *Proc. IEEE Congr. Evol. Comput.*, May 2002, pp. 413–418.
[17] Y. Shi, H. Teng, and Z. Li, "Cooperative co-evolutionary differential evolution for function optimization," in *Proc. 1st Int. Conf. Nat. Comput.*, 2005, pp. 1080–1088.
[18] H. Chen, Y. Zhu, K. Hu, X. He, and B. Niu, "Cooperative approaches to bacterial foraging optimization," in *Proc. ICIC*, 2008, pp. 541–548.
[19] Z. Yang, K. Tang, and X. Yao, "Multilevel cooperative coevolution for large scale optimization," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2008, pp. 1663–1670.
[20] M. Clerc and J. Kennedy, "The particle swarm: Explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.
[21] J. Kennedy, "Bare bones particle swarm," in *Proc. IEEE SIS*, Apr. 2003, pp. 80–87.
[22] T. Richer and T. Blackwell, "The Lévy particle swarm," in *Proc. IEEE CEC*, Sep. 2006, pp. 808–815.
[23] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.
[24] C. Lee and X. Yao, "Evolutionary programming using the mutations based on the Lévy probability distribution," *IEEE Trans. Evol. Comput.*, vol. 8, no. 1, pp. 1–13, Feb. 2004.
[25] B. Secrest and G. Lamont, "Visualizing particle swarm optimization: Gaussian particle swarm optimization," in *Proc. IEEE SIS*, Apr. 2003, pp. 198–204.
[26] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Human Sci.*, Oct. 1995, pp. 39–43.
[27] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proc. IEEE CEC*, May 2002, pp. 1671–1676.
[28] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proc. IEEE SIS*, Apr. 2007, pp. 120–127.
[29] M. Locatelli, "A note on the Griewank test function," *J. Global Optimiz.*, vol. 25, no. 2, pp. 169–174, Feb. 2003.
[30] R. Salomon, "Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions," *BioSyst.*, vol. 39, no. 3, pp. 263–278, 1996.
[31] A. Iorio and X. Li, "Rotated test problems for assessing the performance of multiobjective optimization algorithms," in *Proc. GECCO*, 2006, pp. 683–690.
[32] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, 2001.
[33] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *Proc. IEEE CEC*, Sep. 2005, pp. 1769–1776.
[34] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Posik, "Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009," in *Proc. Workshop GECCO*, 2010, pp. 1689–1696.
[35] R. Ros and N. Hansen, "A simple modification in CMA-ES achieving linear time and space complexity," in *Proc. PPSN X*, Apr. 2008, pp. 296–305.
[36] N. Hansen. *CMA Evolution Strategy Source Code* [Online]. Available: http://www.bionik.tu-berlin.de/user/niko/cmaes_inmatlab.html
[37] K. Tang, "Summary of results on CEC'08 competition on large scale global optimization," Nature Inspired Computat. Applicat. Lab., Univ. Sci. Technol. China, Hefei, China, 2008.
[38] N. Hansen and S. Kern, "Evaluating the CMA evolution strategy on multimodal test functions," in *Proc. PPSN VIII*, 2004, pp. 282–291.
[39] S. Hsieh, T. Sun, C. Liu, and S. Tsai, "Solving large scale global optimization using improved particle swarm optimizer," in *Proc. IEEE CEC*, Jun. 2008, pp. 1777–1784.
[40] S. Zhao, J. Liang, and P. Suganthan, "Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization," in *Proc. IEEE CEC*, Jun. 2008, pp. 3845–3852.
[41] T. Ray and X. Yao, "A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning," in *Proc. IEEE CEC*, May 2009, pp. 983–999.
[42] P. Zhang, X. Yao, L. Jia, B. Sendhoff, and T. Schnier, "Target shape design optimization by evolving splines," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 2009–2016.

Xiaodong Li (M'03–SM'07) received the B.S. degree from Xidian University, Xi'an, China, in 1988, and the Dipl.Com. and Ph.D. degrees from the University of Otago, Dunedin, New Zealand, in 1992 and 1998 respectively, all in information science.

He is currently with the School of Computer Science and Information Technology, Royal Melbourne Institute of Technology, Melbourne, Australia. His current research interests include evolutionary computation (in particular, evolutionary multiobjective optimization, evolutionary optimization in dynamic environments, and multimodal optimization), neural networks, complex systems, and swarm intelligence.

Dr. Li is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and *International Journal of Swarm Intelligence Research*. He is a member of the IEEE Computational Intelligence Society (CIS) Task Force on Swarm Intelligence (since its inception), the IEEE CIS Task Force on Evolutionary Computation in Dynamic and Uncertain Environments, and the IEEE CIS Task Force on Large Scale Global Optimization. He is a member of the Technical Committee on Soft Computing, of the Systems, Man and Cybernetics Society and a member of the IASR Board of Editors for the *Journal of Advanced Research in Evolutionary Algorithms*.

Xin Yao (M'91–SM'96–F'03) received the B.S. degree from the University of Science and Technology of China (USTC), Hefei, China, in 1982, the M.S. degree from the North China Institute of Computing Technology, Beijing, China, in 1985, and the Ph.D. degree from USTC in 1990.

He was an Associate Lecturer and a Lecturer with the USTC, from 1985 to 1990, a Post-Doctoral Fellow with the Australian National University, Canberra, Australia, and CSIRO, Melbourne, Australia, from 1990 to 1992, and a Lecturer, Senior Lecturer, and Associate Professor with the University of New South Wales at the Australian Defence Force Academy, Canberra, from 1992 to 1999. Since April 1999, he has been a Professor (Chair) of computer science with the University of Birmingham, Birmingham, U.K., where he is currently the Director of the Center of Excellence for Research in Computational Intelligence and Applications. He is also a Distinguished Visiting Professor (Grand Master Professorship) of USTC. He has more than 350 refereed publications. His current research interests include evolutionary computation and neural network ensembles.

Dr. Yao was the recipient of the 2001 IEEE Donald G. Fink Prize Paper Award, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding 2008 Paper Award, and several other best paper awards. He is a Distinguished Lecturer of the IEEE Computational Intelligence Society, an invited Keynote/Plenary Speaker of 60 international conferences, and a former, from 2003 to 2008, Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.