A Competitive Swarm Optimizer for Large Scale Optimization

Ran Cheng and Yaochu Jin, *Senior Member, IEEE*

***Abstract*—In this paper, a novel competitive swarm optimizer (CSO) for large scale optimization is proposed. The algorithm is fundamentally inspired by the particle swarm optimization (PSO) but conceptually very different. In the proposed CSO, neither the personal best position of each particle nor the global best position (or neighborhood best positions) is involved in updating the particles. Instead, a pairwise competition mechanism is introduced, where the particle that loses the competition will update its position by learning from the winner. To understand the search behavior of the proposed CSO, a theoretical proof of convergence is provided, together with empirical analysis of its exploration and exploitation abilities showing that the proposed CSO achieves a good balance between exploration and exploitation. Despite its algorithmic simplicity, our empirical results demonstrate that the proposed CSO exhibits better overall performance than five state-of-the-art metaheuristic algorithms on a set of widely used large-scale optimization problems and is able to effectively solve problems of dimensionality up to 5000.**

***Index Terms*—Particle swarm optimization, competition, learning, convergence analysis, competitive swarm optimizer, large scale optimization**

I. Introduction

PARTICLE swarm optimization (PSO) is one powerful and widely used swarm intelligence paradigm [1] introduced by Kennedy and Eberhart in 1995 [2] for solving optimization problems. The algorithm is based on a simple mechanism that mimics swarm behaviors of social animals such as bird flocking. Due to its simplicity in implementation, PSO has witnessed a rapid increase in popularity over the past decades.

PSO contains a swarm of particles, each of which has a position and velocity flying in an n-dimensional search space, representing a candidate solution of the optimization problem to be solved. To locate the global optimum, the velocity and position of each particle are updated iteratively using the following equations:

$$\begin{aligned} V_i(t+1) = \omega V_i(t) &+ c_1 R_1(t)(pbest_i(t) - X_i(t)) \\ &+ c_2 R_2(t)(gbest(t) - X_i(t)), \end{aligned} \tag{1}$$

$$X_i(t+1) = X_i(t) + V_i(t+1), \tag{2}$$

where t is the iteration (generation) number, $V_i(t)$ and $X_i(t)$ represent the velocity and position of the i-th particle, respectively; ω is termed inertia weight [3], c_1 and c_2 are the acceleration coefficients [4], $R_1(t)$ and $R_2(t)$ are two vectors randomly generated within $[0,1]^n$; $pbest_i(t)$ and $gbest(t)$ are the best solution of the i-th particle found so far, often known as the *personal best*, and the best solution found by all particles so far, known as the *global best*, respectively. Kennedy has referred $c_1 R_1(t)(pbest_i(t) - X_i(t))$ and $c_2 R_2(t)(gbest(t) - X_i(t))$ as the *cognitive* component and *social* component, respectively [5].

Due to its conceptual simplicity and high search efficiency, PSO has attracted much research interest over the past decades and has been successfully applied to a number of applications, such as water distribution network design [6], parameter optimization in suspension system [7], resource allocation [8], task assignment [9], DNA sequence compression [10] and many others [11]. However, it has been found that PSO perform poorly when the optimization problem has a large number of local optima or is high-dimensional [12]. The above weaknesses can usually be attributed to the premature convergence that often occurs in PSO [13].

As a typical population-based optimization technique, convergence speed and global search ability are two critical criteria for the performance of PSO algorithms. In order to alleviate premature convergence by achieving a good balance fast convergence and global search ability, a number of PSO variants have been suggested, which can be largely classified into the following four categories [13]:

1) Adaptation of the control parameters. ω, c_1 and c_2 are the three control parameters in the canonical PSO, as shown in (1). ω, termed the inertia weight, was first proposed by Shi and Eberhart to balance between global search and local refinement [14]. The inertia weight ω was further modified by linearly decreasing it from 0.9 to 0.4 over the search procedure [15]. Another important modification of ω is the introduction of fuzzy inference [16].
   Methods for adapting c_1 and c_2 (called the acceleration coefficients) have also been suggested. Time-varying acceleration coefficients were first introduced by Ratnaweera *et al.* in [17]. Similarly, a time-varying PSO based on a novel operator was introduced in [18]. Most recently, a multiple parameter control mechanism has been introduced to adaptively change all the three parameters in [19].
2) Modifications in topological structures. The motivation of introducing topological structures in PSO is to enhance the swarm diversity using neighborhood control [20], [21]. Several topological structures have been proposed [22], including the ring topology and the von Neumann topology. In [23], a fully informed PSO (FIPS)

Ran Cheng and Yaochu Jin are with the Department of Computing, University of Surrey, Guildford, Surrey, GU2 7XH, United Kingdom (e-mail: {r.cheng;yaochu.jin}@surrey.ac.uk).

Manuscript received July 20, 2013; revised xxxx, xxxx.

was developed, where the update of each particle is based on the positions of several neighbors. Another important example is the comprehensive learning PSO (CLPSO) introduced in [24], where particles update each dimension by learning from different local best positions. Recently, a distance-based locally informed particle swarm optimizer was proposed specifically to tackle multimodal problems in [25].

3) Hybridization with other search techniques. Since different search techniques have different strengths, it is a natural idea to hybridize PSO with other search methods. One straightforward idea is to combine PSO with different evolutionary algorithms, such as genetic algorithms [26], differential evolution [27], and ant colony optimization [28], [29]. Another important idea is to integrate PSO with local search techniques [30], [31]. In addition, various search operators based on sociological or biological concepts have also been proposed, such as the niching PSO [32], cultural-based PSO [33] and the aging theory inspired PSO (ALC-PSO) [13]. Other hybrid PSO variants include PSO with Gaussian mutation [34], PSO with chaos [35], orthogonal learning PSO (OLPSO) [36], PSO with a moderate-random-search strategy [37], and a very recently proposed PSO with a periodic mutation strategy and neural networks [38].
4) Multi-swarm PSO. One early work on multi-swarm PSO was reported in [39], where the sub-swarms cooperate to solve large scale optimization problems. In [30], a dynamic multi-swarm PSO (DMS-PSO) algorithm is proposed to dynamically change the neighborhood structure for a higher degree of swarm diversity, even with a very small swarm size. More multi-swarm PSO variants can be found in [40], [41].

Since most PSO variants introduce new mechanisms or additional operators, the enhancement of search performance is often at the cost of increasing the computational complexity. Furthermore, due to the strong influence of the global best position $gbest$ on the convergence speed [39], premature convergence remains a major issue in most existing PSO variants. To take a closer look into the influence of $gbest$ on premature convergence, we rewrite (1) as follows:

$$V_i(t+1) = \omega V_i(t) + \theta_1(p_1 - X_i(t)), \tag{3}$$

where,

$$\begin{aligned} \theta_1 &= c_1 R_1(t) + c_2 R_2(t), \\ p_1 &= \frac{c_1 R_1(t)}{c_1 R_1(t) + c_2 R_2(t)} pbest_i(t) \\ &\quad + \frac{c_2 R_2(t)}{c_1 R_1(t) + c_2 R_2(t)} gbest(t). \end{aligned} \tag{4}$$

From (3), it can be noticed that the difference between p_1 and X_i serves as the main source of diversity. More precisely, the diversity of p_1 itself is generated by the difference between $pbest_i$ and $gbest$, refer to (4). However, in practice, due to the global influence of $gbest$, $pbest_i$ is very likely to have a value similar to or even the same as $gbest$, thus considerably reducing the swarm diverisity. In other words, as a source of diversity, p_1 largely determines how well PSO is able to balance exploration and exploitation in the search procedure. Having noticed this, Mendes and Kennedy proposed a modified PSO, where p_1 is the best particle in the neighborhood of a particle rather than a combination of $gbest$ and $pbest_i$ [23]. Out of similar considerations, Liang *et. al* also introduced a PSO variant without $gbest$ [24], where the update strategy aims to learn from $pbest_i$ only.

In order to address premature convergence, a step further might be to completely get rid of $gbest$ and $pbest_i$. An attempt was made along this line in a multi-swarm framework [42], where neither $gbest$ nor $pbest_i$ has been used. In this multi-swarm framework, the update of particles is driven by a pairwise competition mechanism between particles from the two swarms. After each competition, the loser will be updated according to the information from the winner swarm, while the winner will be updated using a mutation strategy. In the experiments, the framework showed promising performance on relatively high-dimensional problems.

Following the idea in [42], in this work, we explore the use of the competition mechanism between particles within one single swarm. In addition, the particle loses a competition will learn from the winner particle instead of from $gbest$ or $pbest_i$. Since the main driving force behind this idea is the pairwise competition mechanism between different particles and neither $gbest$ nor $pbest_i$ is involved in the search process, we term the proposed algorithm *Competitive Swarm Optimizer (CSO)* to avoid ambiguity. CSO distinguishes itself from the canonical PSO mainly in the following aspects:

1) In the canonical PSO, the dynamic system is driven mostly by the global best position $gbest$ and individual best position $pbest_i$, whilst in CSO, there is no more $gbest$ or $pbest_i$. Instead, the dynamic system is driven by a *random competition* mechanism, where any particle could be a potential leader;
2) In PSO, the historical best positions are recorded, whilst in CSO, there is no memory used to memorize the historical positions. By contrast, the particles that lose a competition learn from the winners in the current swarm only.

The rest of this paper is organized as follows. Section II presents the details of CSO, followed by an empirical analysis of search behaviors and a theoretical convergence proof in Section III. Section IV first presents some statistical results that compare CSO with a few state-of-the-art algorithms on the CEC'08 benchmark functions of dimensionality up to 1000 [43]. Empirical investigations on the influence of the parameter settings are also conducted. The search ability of CSO has been challenged further with the test functions of 2000 and 5000 dimensions, which, to the best of our knowledge, are the highest dimensions that have ever been reported in the evolutionary optimization literature. Finally, the influence of neighborhood control on the search performance has been investigated. Conclusions will be drawn in Section V.

II. ALGORITHM

Without loss of generality, we consider the following minimization problem:

$$\begin{aligned} &\min f = f(X) \\ &\text{s.t.} \quad X \in \mathcal{X} \end{aligned} \tag{5}$$

where $\mathcal{X} \in \mathbb{R}^n$ is the feasible solution set, n denotes the dimension of the search space, i.e., the number of decision variables.

In order to solve the optimization problem above, a swarm $P(t)$ contains m particles is randomly initialized and iteratively updated, where m is known as the swarm size and t is the generation index. Each particle has an n-dimensional position, $X_i(t) = (x_{i,1}(t), x_{i,2}(t), ..., x_{i,n}(t))$, representing a candidate solution to the above optimization problem, and an n-dimensional velocity vector, $V_i(t) = (v_{i,1}(t), v_{i,2}(t), ..., v_{i,n}(t))$. In each generation, the particles in $P(t)$ are randomly allocated into $m/2$ couples (assuming that the swarm size m is an even number), and afterwards, a competition is made between the two particles in each couple. As a result of each competition, the particle having a better fitness, hereafter denoted as *winner*, will be passed directly to the next generation of the swarm, $P(t+1)$, while the particle that loses the competition, the *loser*, will update its position and velocity by learning from the winner. After learning from the winner, the loser will also be passed to swarm $P(t+1)$. This means that each particle will participate in a competition only once. In other words, for a swarm size of m, $m/2$ competitions occur so that all m particles participate in one competition once and the position and velocity of $m/2$ particles will be updated. Fig. 1 illustrates the main idea of CSO.

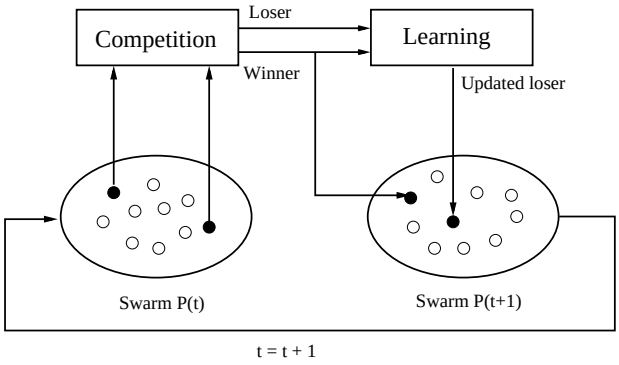


Fig. 1. The general idea of CSO. During each generation, particles are pairwise randomly selected from the current swarm for competitions. After each competition, the loser, whose fitness value is worse, will be updated by learning from the winner, while the winner is directly passed to the swarm of the next generation.

Let us denote the position and velocity of the winner and loser in the k-th round of competition in generation t with $X_{w,k}(t)$, $X_{l,k}(t)$, and $V_{w,k}(t)$, $V_{l,k}(t)$, respectively, where $k = 1, 2, ..., m/2$. Accordingly, after the k-th competition the loser's velocity will be updated using the following learning strategy:

$$\begin{aligned} V_{l,k}(t+1) = &R_1(k,t)V_{l,k}(t) \\ &+ R_2(k,t)(X_{w,k}(t) - X_{l,k}(t)) \\ &+ \varphi R_3(k,t)(\bar{X}_k(t) - X_{l,k}(t)). \end{aligned} \tag{6}$$

As a result, the position of the loser can be updated with the new velocity:

$$X_{l,k}(t+1) = X_{l,k}(t) + V_{l,k}(t+1), \tag{7}$$

where $R_1(k,t), R_2(k,t), R_3(k,t) \in [0,1]^n$ are three randomly generated vectors after the k-th competition and learning process in generation t, $\bar{X}_k(t)$ is the mean position value of the relevant particles, φ is a parameter that controls the influence of $\bar{X}(t)$. Specifically, for $\bar{X}_k(t)$, a global version and a local version can be adopted:

1) $\bar{X}_k^g(t)$ denotes the global mean position of all particles in $P(t)$;
2) $\bar{X}_{l,k}^l(t)$ means the local mean position of the particles in a predefined neighborhood of particle l.

It has been found that neighborhood control is able to help improve PSO's performance on multimodal function by maintaining a higher degree of swarm diversity [21]. Similarly, the motivation to introduce neighborhood control in $\bar{X}_k(t)$ is to increase swarm diversity, which potentially enhances the search performance of CSO. In the remainder of this paper, the global version $\bar{X}_k^g(t)$ is adopted as a default setup unless otherwise specified. The performance of CSO using the local version $\bar{X}_{l,k}^l(t)$ will be investigated in Section IV.D.

In order to gain a better understanding of the learning strategy in CSO, we provide below more discussions about (6).

1) The first part $R_1(k,t)V_{l,k}(t)$ is similar to the *inertia term* in the canonical PSO, refer to (1), which ensures the stability of the search process. The only difference is that the inertia weight ω in PSO is replaced by a random vector $R_1(t)$ in CSO.
2) The second part $R_2(k,t)(X_{w,k}(t) - X_{l,k}(t))$ is also called *cognitive component* after Kennedy and Eberhart. Different from the canonical PSO, the particle that loses the competition learns from its competitor, instead of from its personal best position found so far. This mechanism may be biologically more plausible in simulating animal swarm behaviors, since it is hard to require that all particles memorize the best position they have experienced in the past.
3) The third part $\varphi R_3(k,t)(\bar{X}(t) - X_{l,k)}(t))$ is termed *social component*, again after Kennedy and Eberhart. However, the particle that loses the competition learns from the mean position of the current swarm rather than the $gbest$, which requires no memory and makes good sense, biologically.

With the descriptions and definitions above, the pseudo code of CSO algorithm can be summarized in Algorithm 1. We can see that CSO maintains the algorithmic simplicity of PSO, which is quite different from most existing PSO variants. Apart from the fitness evaluations, which is problem dependent [44], [45], the main computational cost in CSO is

Algorithm 1 The pseudocode of the Competitive Swarm Optimizer (CSO). t is the generation number. U denotes a set of particles that have not yet participated in a competition. Unless otherwise specified, the *terminal condition* is the maximum number of fitness evaluations.

```
t = 0;
randomly initialize P(0);
while terminal condition is not satisfied do
    calculate the fitness of all particles in P(t);
    U = P(t), P(t+1) = ∅;
    while U ≠ ∅ do
        randomly choose two particles X_1(t), X_2(t) from U;
        if f(X_1(t)) ≤ f(X_2(t)) then
            X_w(t) = X_1(t), X_l(t) = X_2(t);
        else
            X_w(t) = X_2(t), X_l(t) = X_1(t);
        end if
        add X_w(t) into P(t+1);
        update X_l(t) using (6) and (7);
        add the updated X_l(t+1) to P(t+1);
        remove X_1(t), X_2(t) from U;
    end while
    t = t + 1;
end while
```

the update of $X_l(t)$, which is an inevitable operation in most swarm or population based evolutionary algorithms [2], [46]–[48]. Consequently, the computational complexity of CSO is $O(mn)$, where m is the swarm size and n is the search dimensionality.

III. Search Dynamics Analysis and Convergence Proof

In order to better understand the search mechanism in CSO, we will carry out empirical studies on its search dynamics by comparing it with the canonical PSO. In addition, a theoretical proof of convergence will be given, which shows that CSO, similar to the canonical PSO, will converge to an equilibrium. Note, however, that this equilibrium is not necessarily the global optimum.

A. Analysis of search dynamics

1) Exploration: Exploration is desirable in the early stage of optimization to perform global search and locate the optimum regions. To examine the exploration ability of CSO, we reformulate (6) into a form similar to (3):

$$V_i(t+1) = R_1(k,t)V_i(t) + \theta_2(p_2 - X_i(t)), \tag{8}$$

then the following expression can be obtained:

$$\begin{aligned}\theta_2 &= R_2(k,t) + \varphi R_3(k,t)\\ p_2 &= \frac{R_2(k,t)}{R_2(k,t)+\varphi R_3(k,t)}X_w(t) + \frac{\varphi R_3(k,t)}{R_2(t)+\varphi R_3(k,t)}\bar{X}(t).\end{aligned} \tag{9}$$

Compared to (4), it can be observed that (9) has better chance to generate a higher degree of diversity. On the one hand, particle $X_w(t)$ is randomly chosen from the swarm before the competition, whilst $gbest(t)$ is deterministically updated and shared by all particles, and $pbest_i(t)$ is also deterministically updated and always used by particle i. On the other hand, although $\bar{X}(t)$ is shared by several particles, it depends on the mean current position of the whole swarm, which will be less likely to introduce bias towards any particular particle. Finally, it is noted that in CSO, only half of the particles will be updated in each generation, while in PSO, all particles are updated.

To illustrate the above intuitive observation, three typical cases are considered below to show how CSO is potentially able to perform more explorative search than the canonical PSO.

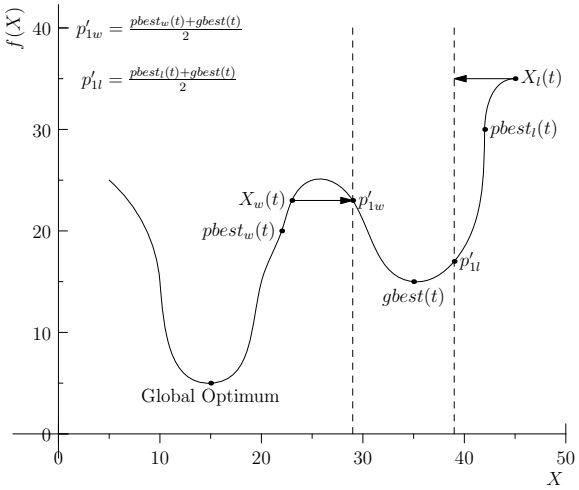


Fig. 2. Illustration of the search dynamics of the canonical PSO on a multimodal optimization problem. In this case, $gbest(t)$ is in a local optimum region and the two particles $w(t)$ and $l(t)$ is attracted into this region.

Let us first consider a situation where $gbest(t)$ is trapped in a local optimum, as illustrated in Fig. 2. For particle $l(t)$, since both $X_l(t)$ and $pbest_l(t)$ are located inside the local optimum region, the particle will move towards the local optimum position recorded by $gbest(t)$. By contrast, although both $X_l(t)$ and $pbest_l(t)$ are located outside the optimum region, particle $w(t)$ will still move in a wrong direction towards the local optimum due to the dynamics driven by $gbest(t)$.

A natural idea to avoid the situation shown in Fig. 2 is to remove the $gbest(t)$ from the update strategy so that particles will learn from $pbest(t)$ only. This methodology has already been adopted by Liang *et al* [24]. In this way, particle $l(t)$ is able to *fly over* the local optimum, refer to Fig. 3. Without $gbest(t)$, PSO seems to be in a better position to perform exploration.

However, although $gbest(t)$ is removed, $pbest(t)$ can still attract the particles into a local optimum region, limiting its ability to explore the whole landscape. Let us consider the situation shown in Fig. 4. In iteration t, both particles reside inside the local optimum region, including their $pbest(t)$. In iteration $t+1$, coincidentally, particle $w(t+1)$ manages to move outside the local optimum region and its new position is $X_w(t+1)$. However, since the fitness value of $pbest_w(t+1)$ is still better than $X_w(t+1)$, $pbest_w(t)$ is not updated. As

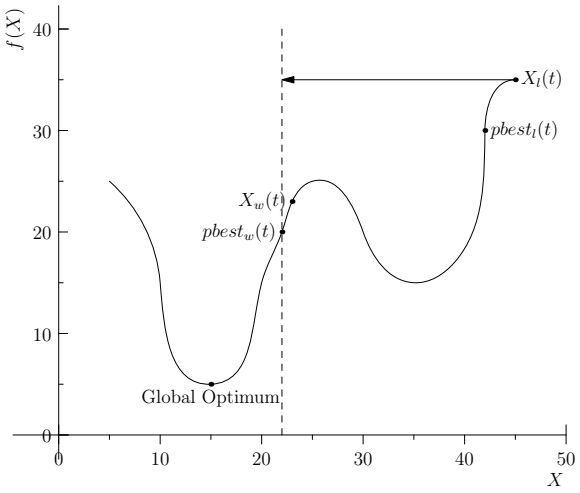


Fig. 3. Illustration of a situation where search is performed in a multimodal space using a PSO variant without $gbest$. In this case, particle l will fly over the global optimum region due to the attraction of the $pbest_w$.

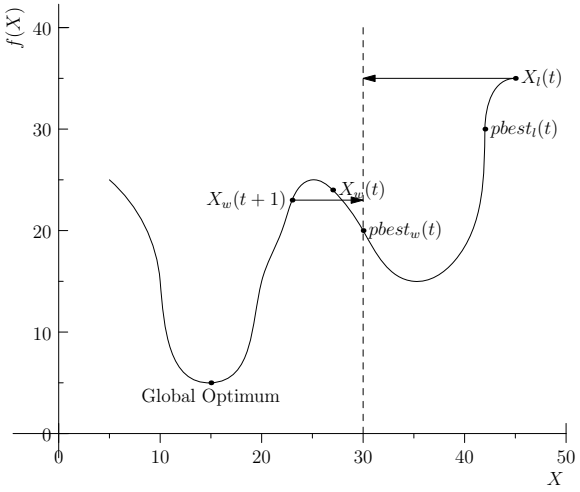


Fig. 4. Illustration of a situation where search is performed in a multimodal space using a PSO variant without $gbest(t)$. In this case, the $pbest_w(t)$ of particle $w(t)$ is located in a local optimum region, serving as a *local* $gbest$. As a result, both particles $l(t)$ and $w(t)$ will be attracted by $pbest_w(t)$.

a consequence, $pbest_w(t)$ continues to serve as an *attractor*, which will pull particle $w(t+1)$ back into the local optimum region again. In this situation, $pbest_w(t)$ has played the role of a *local* $gbest$, although no $gbest(t)$ is adopted. In comparison, both the situations as shown in Fig. 2 and Fig. 4 can be avoided by CSO because both $gbest(t)$ and $pbest(t)$ are removed, refer to Fig. 5.

2) Exploitation: Exploitation is required in the later search stage to refine the solution found at the exploration stage. To analyze the exploitation behavior of CSO, we randomly pick up two particles w and l from the swarm and the following relationship holds:

$$f(X_w(t)) \leq f(X_l(t)), \tag{10}$$

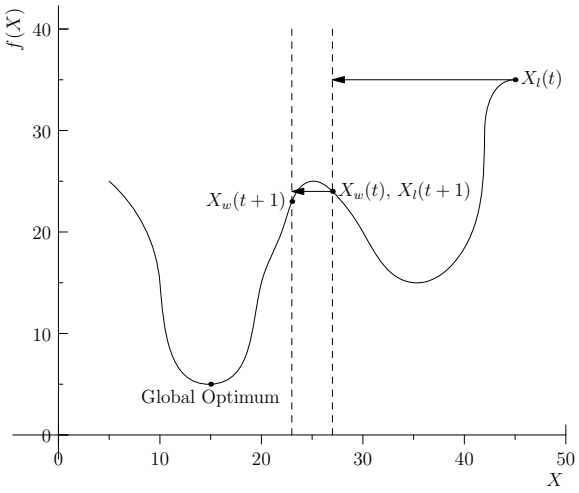


Fig. 5. Illustration of a situation where search is performed in a multimodal space using a PSO variant with neither $gbest(t)$ nor $pbest(t)$ (CSO). In this case, particle $l(t)$ is only attracted by particle $w(t)$, thereby flying over the local optimum region.

According to the definitions of $gbest$ and $pbest$ in the canonical PSO, the following relationship can be obtained:

$$\begin{cases} f(gbest(t)) \leq f(pbest_w(t)) \leq f(X_w(t)), \\ f(gbest(t)) \leq f(gbest_l(t)) \leq f(X_l(t)). \end{cases} \tag{11}$$

When t becomes very large in the late search stage, the following relationship between $pbest_w(t)$, $pbest_l(t)$ and $gbest(t)$ holds:

$$\begin{cases} pbest_w(t) \approx gbest(t), \\ pbest_l(t) \approx gbest(t) \end{cases}. \tag{12}$$

Let

$$\begin{aligned} \Delta F_1(t) &= |f(X_l(t)) - f(gbest)| \\ &= |f(X_l(t)) - f(\frac{gbest(t) + gbest(t)}{2})| \\ &\approx |f(X_l(t)) - f(\frac{gbest(t) + pbest_l(t)}{2})| \\ &= |f(X_l(t)) - f(p_1')|, \\ \Delta F_2(t) &= |f(X_l(t)) - f(X_w(t))| \\ &= |f(X_l(t)) - f(p_2')|, \end{aligned} \tag{13}$$

where p_1' is the expected value of p_1 in (4) and p_2' is the expected value of p_2 in (9) with $\varphi = 0$. Then the following relationship can be obtained from (10), (11) and (12):

$$\Delta F_2(t) \leq \Delta F_1(t). \tag{14}$$

The relationship in (14) indicates that CSO, in comparison with the canonical PSO, has a better ability to exploit the small gaps between two positions whose fitness values are very similar.

B. Theoretical convergence proof

Similar to most theoretical convergence analysis of PSO [49]–[51], a deterministic implementation of CSO is considered to theoretically analyze its convergence property. It should

also be pointed out that the proof does not guarantee the convergence to the global optimum.

For any particle i in $P(t)$, it can have the following two possible behaviors after participating in a competition:

1) $X_i(t+1) = X_i(t)$, if $X_i(t)$ is a winner;
2) $X_i(t+1)$ is updated using (6) and (7), if $X_i(t)$ is a loser.

In case $X_i(t)$ is a winner, the particle will not be updated. Therefore, we only need to consider the case when $X_i(t)$ is a loser and then updated. Without loss of generality, we can rewrite (6) and (7) by considering an one-dimension deterministic case:

$$\begin{aligned} V_i(t+1) &= \frac{1}{2}V_i(t) \\ &\quad + \frac{1}{2}(X_w(t) - X_i(t)) \\ &\quad + \varphi\frac{1}{2}(\bar{X}(t) - X_i(t)), \\ X_i(t+1) &= X_i(t) + V_i(t+1), \end{aligned} \tag{15}$$

where $\frac{1}{2}$ is the expected value of R_1, R_2 and R_3, $X_w(t)$ is the position of the winner in the competition with the i-th particle.

Theorem 1. *For any given $\varphi \geq 0$, the dynamic system described by (15) will converge to an equilibrium.*

Proof. Let

$$\begin{aligned} \theta &= \frac{1+\varphi}{2}, \\ p &= \frac{1}{1+\varphi}X_w(t) + \frac{\varphi}{1+\varphi}\bar{X}(t), \end{aligned} \tag{16}$$

then (15) can be simplified to:

$$\begin{aligned} V_i(t+1) &= \frac{1}{2}V_i(t) + \theta(p - X_i(t)) \\ X_i(t+1) &= X_i(t) + V_i(t+1), \end{aligned} \tag{17}$$

The search dynamics described in (17) can be seen as a dynamical system, and the convergence analysis of the system can be conducted by using the well-established theories on stability analysis in dynamical systems. To this end, we rewrite system (17) in the following form:

$$y(t+1) = Ay(t) + Bp, \tag{18}$$

where

$$y(t) = \begin{bmatrix} V_i(t) \\ X(t) \end{bmatrix}, A = \begin{bmatrix} \frac{1}{2} & -\theta \\ \frac{1}{2} & 1-\theta \end{bmatrix}, B = \begin{bmatrix} \theta \\ \theta \end{bmatrix}, \tag{19}$$

where A is called *state matrix* in dynamical system theory, p is called *external input* that drives the particle to a specific position and B is called *input matrix* that controls external effect on the dynamics of the particle.

If there exists an *equilibrium* y^* that satisfies $y^*(t+1) = y^*(t)$ for any t, it can be calculated from (18) and (19):

$$y^* = \begin{bmatrix} 0 & p \end{bmatrix}, \tag{20}$$

which means that the particles will finally stabilize at the same position, provided that p is constant, i.e., an optimum (local or global) has been found, so that no more update for p will happen.

Convergence means that the particles will eventually settle down at the equilibrium point y^*. From the dynamical system theory, we can know that the convergence property depends on the eigenvalues of the state matrix A:

$$\lambda^2 - (\frac{3}{2} - \theta)\lambda + \frac{1}{2} = 0, \tag{21}$$

where the eigenvalues are:

$$\begin{cases} \lambda_1 = \frac{3}{4} - \frac{\theta}{2} + \frac{\sqrt{(\frac{3}{2}-\theta)^2-2}}{2} \\ \lambda_2 = \frac{3}{4} - \frac{\theta}{2} - \frac{\sqrt{(\frac{3}{2}-\theta)^2-2}}{2} \end{cases} \tag{22}$$

The necessary and sufficient condition for the convergence, i.e., the equilibrium point is a stable attractor, is that $|\lambda_1| < 1$ and $|\lambda_2| < 1$, leading to the result:

$$\theta > 0, \tag{23}$$

where if θ is substituted by φ using (16), the condition for convergence on φ is:

$$\varphi > -1. \tag{24}$$

Therefore, $\varphi \geq 0$ is a sufficient condition for the convergence of the system. □

From Theorem 1, we can conclude that the algorithm will converge to an equilibrium regardless the exact value of φ, as long as $\varphi > -1$. In this work, only non-negative $\varphi \geq 0$ will be adopted.

IV. Experimental studies

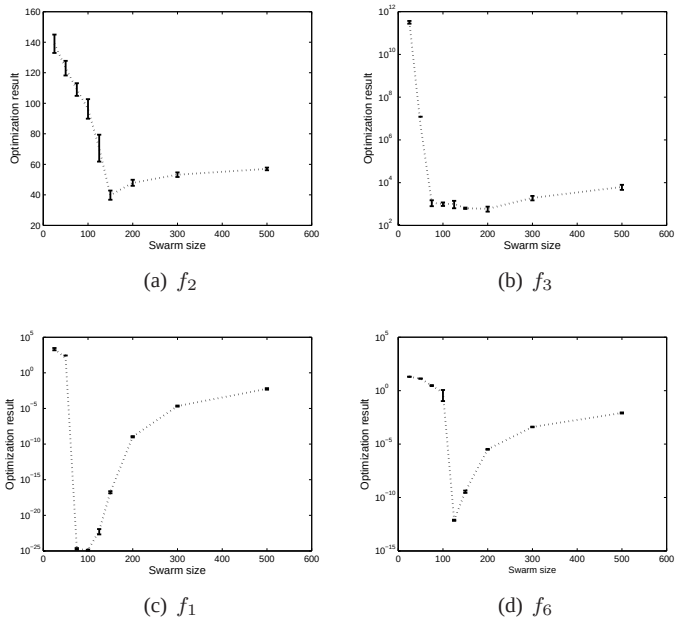
In this section, we will perform a set of experiments conducted on the seven benchmark functions proposed in the CEC'08 special session on large scale optimization problems [43]. Among the seven functions, f_1 (Shifted Sphere), f_4 (Shifted Rastrigin) and f_6 (Shifted Ackley) are separable functions, while the other four functions f_2 (Schwefel Problem), f_3 (Shifted Rosenbrock), f_5 (Shifted Griewank) and f_7 (Fast Fractal) are non-separable. Note that f_5 becomes more separable as the dimension increases, because the product component of f_5 becomes increasingly less significant [52] with an increasing dimension. Therefore, in the following experiments, if the dimension is equal to or higher than 500, f_5 will be regarded a separable function.

At first, experiments are conducted to empirically understand the influence of the two parameters in CSO, namely, the swarm size m and the social factor φ. Then, CSO is compared with a few recently proposed algorithms for large scale optimization on 100-D, 500-D, and 1000-D benchmark functions. Afterwards, regarding the scalability to search dimensions, CSO is further challenged on 2000-D and 5000-D functions. Finally, the influence of neighborhood control on CSO's swarm diversity and search performance is investigated.

The experiments are implemented on a PC with an Intel Core i5-2500 3.3GHz CPU and Microsoft Windows 7 Enterprise SP1 64-bit operating system, and CSO is written in language C on Microsoft Visual Studio 2010 Enterprise.

All statistical results, unless otherwise specified, are averaged over 25 independent runs. For each independent run, the maximum number of fitness evaluations (FEs), as recommended in [43], is set to $5000 * n$, where n is the search dimension of the test functions. In the comparisons between different statistical results, two-tailed t-tests are conducted at a significance level of $\alpha = 0.05$.

A. Parameter settings



(a) f_2 (b) f_3

(c) f_1 (d) f_6

Fig. 6. Statistical results of optimization errors obtained CSO on 2 non-separable functions f_2, f_3 and 2 separable functions f_1, f_6 of 500 dimensions with different swarm sizes m varying from 25 to 300.

1) Swarm size: Like most swarm optimization algorithms, the swarm size is an indispensable parameter. With a small swarm size, the particles tend to converge very fast before the search space is well explored, thus leading to premature convergence; however, if the swarm size is too big, a large number of FEs will be required during each generation, which may become impractical for computationally expensive problems.

Generally, the swarm size is empirically specified. For example, in CCPSO2 [52], multi-swarms were adopted with a swarm size 30 for each swarm, and for 500-D functions, the number of swarms varied from 4 to 50, creating an average size around 240; in DMS-PSO [30], a larger swarm size 450 was adopted for the optimization of 500-D functions.

To gain empirical insight into the influence of the swarm size on the search performance of CSO, the swarm size has been varied from 25 to 300 for the four CEC'08 functions f_1, f_2, f_3 and f_6 of search dimension 500. Among the four functions, f_1 and f_6 are separable and the other two are non-separable. To remove the influence of the *social component*, φ is set to 0 in this set of experiments.

Fig. 6 shows the statistical results of the optimization errors obtained by CSO with different swarm sizes m. It can be seen that CSO performs well on 500-D functions with a swarm size around 150, which is smaller than the swarm sizes adopted in CCPSO2, DMS-PSO and other PSO variants, though CSO is a single-swarm algorithm without any *ad hoc* mechanism for large scale optimization. More interestingly, when the swarm size is bigger than some specific values (e.g. 100 for f_1), the optimization performance begin to deteriorate. The reason is that with a bigger swarm size, more FEs (fitness evaluations) have to be performed in each generation. Since the terminal condition in this experiment is the maximal number of FEs, a larger swarm size means a smaller number of generations. This also implies that the performance of CSO does not rely much on a large swarm size. From Fig. 6, we can also see that a swarm size smaller than 100 might be too small for 500-dimensional problems. Based on the observations and discussions above, the swarm size should not be smaller than 200 for real-world large scale ($D \geq 500$) optimization problems.

2) Social factor: In the following, we investigate the influence of the *social component* by varying the *social factor* φ. To this end, simulations have been conducted on the four functions with the swarm size m varying from 200 to 1000 and φ varying from 0 to 0.3.

From the statistical results summarized in Table I, we can see that the best statistical results are the diagonal elements in the table, which implies that there exist a correlation between m and φ. Additionally, it can also be noticed that the non-separable functions (f_2 and f_3) require a smaller φ than the separable functions (f_1 and f_6) to achieve good performance. The reason might be that separable functions are easier to optimize, as a result, a bigger φ would work better because it leads to faster convergence. The best combinations observed from Table I are summarized in Table II.

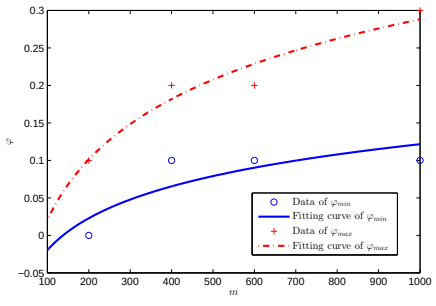


Fig. 7. Fitting curves describing the relationship between the social factor ϕ and swarm size m that lead to the best search performance using the logarithmic linear regression analysis.

For a deep insight into the relationship between the optimal pair of φ and m, a logarithmic linear regression analysis has been performed to model the relationship between m and φ using the data in Table II, as shown in Fig. 7. Based on the regression analysis result, the following empirical setups for φ and swarm size m is recommended:

$$\begin{cases} \varphi(m) = 0 & \text{if } m \leq 100, \\ \varphi(m) \in [\varphi_L(m), \varphi_U(m)] & \text{otherwise} \end{cases} \tag{25}$$

where $\varphi_L(m)$ and $\varphi_L(m)$ are the lower and upper bound

TABLE I
STATISTICAL RESULTS (MEAN VALUES AND STANDARD DEVIATIONS) OF OPTIMIZATION ERRORS OBTAINED BY CSO ON 2 NON-SEPARABLE FUNCTIONS f_2, f_3 AND 2 SEPARABLE FUNCTIONS f_1, f_6 OF 500 DIMENSIONS WITH THE SWARM SIZE m VARYING FROM 200 TO 1000 AND φ FROM 0 TO 0.3.

Swarm size	Function	$\varphi = 0$	$\varphi = 0.1$	$\varphi = 0.2$	$\varphi = 0.3$
$m = 200$	f_2	**4.79E+01(1.97E+00)**	8.26E+01(2.85E+00)	8.58E+01(3.48E+00)	8.45E+01(1.27E+00)
	f_3	**5.95E+02(1.55E+02)**	8.25E+02(5.28E+01)	1.07E+07(4.57E+06)	4.33E+09(6.06E+08)
	f_1	1.08E-09(2.26E-10)	**4.73E-23(8.70E-25)**	1.51E+01(1.77E+01)	3.50E+04(3.77E+03)
	f_6	3.24E-06(2.96E-07)	**3.57E-13(1.02E-14)**	2.78E+00(1.75E-01)	1.04E+01(7.18E-01)
$m = 400$	f_2	6.09E+01(1.06E+00)	**5.47E+01(3.46E+00)**	7.41E+01(7.92E-01)	6.09E+01(1.06E+00)
	f_3	1.31E+06(1.22E+05)	**4.90E+02(1.27E-01)**	5.01E+03(6.21E+02)	2.75E+08(3.96E+07)
	f_1	3.17E+00(3.86E-01)	4.38E-16(5.76E-17)	**3.22E-22(2.41E-23)**	1.29E+03(2.69E+02)
	f_6	2.94E-01(1.76E-02)	1.49E-09(4.36E-11)	**8.82E-13(1.42E-14)**	4.00E+00(1.85E-01)
$m = 600$	f_2	6.52E+01(7.68E-01)	**2.74E+01(3.76E+00)**	6.72E+01(1.26E+00)	7.17E+01(8.32E-01)
	f_3	2.75E+08(3.96E+07)	**4.92E+02(4.00E-01)**	1.39E+03(3.84E+02)	5.28E+07(1.09E+07)
	f_1	3.26E+02(2.84E+01)	2.57E-08(1.69E-09)	**5.46E-22(2.72E-23)**	3.73E+01(1.61E+01)
	f_6	3.33E+00(3.07E-02)	1.27E-05(6.24E-07)	**1.10E-12(1.48E-14)**	1.91E+00(8.08E-02)
$m = 1000$	f_2	7.12E+01(7.47E-01)	**3.22E+01(4.68E-01)**	6.11E+01(1.61E+00)	6.89E+01(7.57E-01)
	f_3	1.40E+09(1.26E+08)	6.43E+02(1.68E+01)	**5.97E+02(6.90E+01)**	8.34E+06(1.76E+06)
	f_1	5.74E+03(4.94E+02)	1.19E-02(5.35E-04)	7.26E-12(2.89E-13)	**2.01E-18(7.69E-19)**
	f_6	6.75E+00(4.61E-02)	9.58E-03(3.70E-04)	1.60E-07(8.03E-09)	**2.55E-11(5.88E-12)**

Two-tailed t-tests have been conducted between the statistical results in each row. If one result is significantly better than all the other errors, it is highlighted. Note that the statistical results are shown in the order of f_2, f_3 (non-separable functions) and f_1, f_6 (separable functions), to clearly see the different φ values for non-separable and separable functions.

of the recommended social factor that can be determined as follows:

$$\begin{cases} \varphi_L(m) = 0.14\log(m) - 0.30 \\ \varphi_R(m) = 0.27\log(m) - 0.51 \\ \varphi_L(m), \varphi_R(m) \geq 0 \end{cases}. \quad (26)$$

TABLE II
THE BEST COMBINATIONS OF THE SWARM SIZE m AND THE SOCIAL FACTOR φ IN THE OPTIMIZATION OF 500-D f_1, f_2, f_3 AND f_6.

	$m = 200$	$m = 400$	$m = 600$	$m = 1000$
φ_{min}	0	0.1	0.1	0.1
φ_{max}	0.1	0.2	0.2	0.3

φ_{max} and φ_{min} denote the maximal and minimal φ that perform best with corresponding m, respectively.

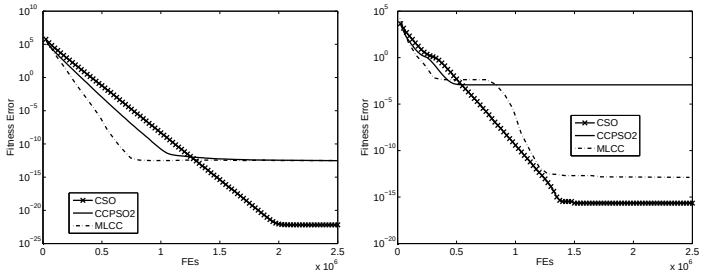
TABLE III
PARAMETER SETTINGS FOR THE SEVEN FUNCTIONS OF 100-D, 500-D AND 1000-D.

Parameter	Dimensions	Separable functions	Non-separable functions
m	100-D	100	100
	500-D	250	250
	1000-D	500	500
φ	100-D	0	0
	500-D	0.1	0.05
	1000-D	0.15	0.1

Separable functions include f_1, f_4, f_5, f_6. Non-separable functions include f_2,f_3, f_7. Note that f_5 is grouped as a non-separable function because the product component becomes less significant with the increase of dimension [52].

B. Benchmark comparisons

In order to verify the performance of CSO for large scale optimization, CSO has been compared with a number of the state-of-the-art algorithms tailored for large scale optimization on the CEC'08 test functions with dimensions of 100, 500 and 1000. The compared algorithms for large scale optimization include CCPSO2 [52], MLCC [53], sep-CMA-ES [54], EPUS-PSO [55] and DMS-PSO [30]. The same criteria proposed in the CEC'08 special session on large scale optimization [43] have been adopted.



(a) f_1 (b) f_5

Fig. 8. The convergence profiles of CSO, CCPSO2 and MLCC on 500-D f_1 and f_5

Among the compared algorithm, the CCPSO2 [52] and the MLCC [53] are designed in the cooperative coevolution (CC) framework [56], which has been proposed to solve high-dimensional problems by automatically implementing the *divide-and-conquer* strategy [57]. Specifically, in both algorithms, random grouping technique is used to divide the whole decision vector into different subcomponents, each of which is solved independently. In CCPSO2, a modified PSO using Cauchy and Gaussian distributions for sampling around the personal best and the neighborhood best positions is adopted as the core algorithm to evolve the CC framework whilst in MLCC, a self-adaptive neighborhood search differential evolution (SaNSDE) is adopted.

The sep-CMA-ES is a simple modification of the original CMA-ES algorithm [58], which has been shown to be more efficient, and to scale surprisingly well on some high-dimensional test functions up to 1000 dimensions [54]. EPUS-PSO and DMS-PSO are another two PSO variants, where the former adjusts the population size according to the search results [55] and the latter adopts a dynamically changing neighborhood structure for each particle [30], and both of them

TABLE IV
THE STATISTICAL RESULTS (FIRST LINE) AND THE t VALUES (SECOND LINE) OF OPTIMIZATION ERRORS ON 100-D TEST FUNCTIONS.

100-D	CSO	CCPSO2	MLCC	sep-CMA-ES	EPUS-PSO	DMS-PSO
f_1	9.11E-29(1.10E-28) –	7.73E-14 (3.23E-14) **-1.20E+01**	6.82E-14 (2.32E-14) **-1.47E+01**	9.02E-15 (5.53E-15) **-8.16E+00**	7.47E-01 (1.70E-01) **-2.20E+01**	0.00E+00 (0.00E+00) 4.14E+00
f_2	3.35E+01(5.38E+00) –	6.08E+00 (7.83E+00) 1.44E+01	2.53E+01 (8.73E+00) 4.00E+00	2.31E+01 (1.39E+01) 3.49E+00	1.86E+01 (2.26E+0) 1.28E+01	3.65E+00 (7.30E-01) 2.75E+01
f_3	3.90E+02(5.53E+02) –	4.23E+02 (8.65E+02) -1.61E-01	1.50E+02 (5.72E+01) 2.16E+00	4.31E+00 (1.26E+01) 3.49E+00	4.99E+03 (5.35E+03) **-4.28E+00**	2.83E+02 (9.40E+02) 4.91E-01
f_4	5.60E+01(7.48E+00) –	3.98E-02 (1.99E-01) 3.74E+01	4.39E-13 (9.21E-14) 3.74E+01	2.78E+02 (3.43E+01) **-3.16E+01**	4.71E+02 (5.94E+01) **-3.47E+01**	1.83E+02 (2.16E+01) **-2.78E+01**
f_5	0.00E+00(0.00E+00) –	3.45E-03 (4.88E-03) **-3.53E+00**	3.41E-14 (1.16E-14) **-1.47E+01**	2.96E-04 (1.48E-03) -1.00E+00	3.72E-01 (5.60E-02) **-3.32E+01**	0.00E+00 (0.00E+00) 0.00E+00
f_6	1.20E-014(1.52E-015) –	1.44E-13 (3.06E-14) **-2.15E+01**	1.11E-13 (7.87E-15) **-6.18E+01**	2.12E+01 (4.02E-01) **-2.64E+02**	2.06E+0 (4.40E-01) **-2.34E+01**	0.00E+00 (0.00E+00) 3.95E+01
f_7	-7.28E+05(1.88E+04) –	-1.50E+03 (1.04E+01) **-1.93E+02**	-1.54E+03 (2.52E+00) **-1.93E+02**	-1.39E+03 (2.64E+01) **-1.93E+02**	-8.55E+02 (1.35E+01) **-1.93E+02**	-1.14E+03 (8.48E+00) **-1.93E+02**
$w/t/l$	–	4/1/2	4/0/3	4/1/2	6/0/1	2/2/3

TABLE V
THE STATISTICAL RESULTS (FIRST LINE) AND THE t VALUES (SECOND LINE) OF OPTIMIZATION ERRORS ON 500-D TEST FUNCTIONS.

500-D	CSO	CCPSO2	MLCC	sep-CMA-ES	EPUS-PSO	DMS-PSO
f_1	6.57E-23(3.90E-24) –	7.73E-14 (3.23E-14) **-1.20E+01**	4.30E-13 (3.31E-14) **-6.50E+01**	2.25E-14 (6.10E-15) **-1.84E+01**	8.45E+01 (6.40E+00) **-6.60E+01**	0.00E+00 (0.00E+00) 8.42E+01
f_2	2.60E+01(2.40E+00) –	5.79E+01 (4.21E+01) **-3.78E+00**	6.67E+01 (5.70E+00) **-3.29E+01**	2.12E+02 (1.74E+01) **-5.29E+01**	4.35E+01 (5.51E-01) **-3.55E+01**	6.89E+01 (2.01E+00) **-6.85E+01**
f_3	5.74E+02(1.67E+02) –	7.24E+02 (1.54E+02) **-3.30E+00**	9.25E+02 (1.73E+02) **-7.30E+00**	2.93E+02 (3.59E+01) 8.23E+00	5.77E+04 (8.04E+03) **-3.55E+01**	4.67E+07 (5.87E+06) **-3.98E+01**
f_4	3.19E+02(2.16E+01) –	3.98E-02 (1.99E-01) 7.38E+01	1.79E-11 (6.31E-11) 7.38E+01	2.18E+03 (1.51E+02) **-6.10E+01**	3.49E+03 (1.12E+02) **-1.39E+02**	1.61E+03 (1.04E+02) **-6.08E+01**
f_5	2.22E-16(0.00E+00) –	1.18E-03 (4.61E-03) -1.28E+00	2.13E-13 (2.48E-14) **-4.29E+01**	7.88E-04 (2.82E-03) -1.40E+00	1.64E+00 (4.69E-02) **-1.75E+02**	0.00E+00 (0.00E+00) 7.85E+84
f_6	4.13E-13(1.10E-14) –	5.34E-13 (8.61E-14) **-6.97E+00**	5.34E-13 (7.01E-14) **-8.53E+00**	2.15E+01 (3.10E-01) **-3.47E+02**	6.64E+00 (4.49E-01) **-7.39E+01**	2.00E+00 (9.66E-02) **-1.04E+02**
f_7	-1.97E+06(4.08E+04) –	-7.23E+03 (4.16E+01) **-2.41E+02**	-7.43E+03 (8.03E+00) **-2.41E+02**	-6.37E+03 (7.59E+01) **-2.41E+02**	-3.51E+03 (2.10E+01) **-2.41E+02**	-4.20E+03 (1.29E+01) **-2.41E+02**
$w/t/l$	–	5/1/1	6/0/1	5/1/1	7/0/0	5/0/2

TABLE VI
THE STATISTICAL RESULTS (FIRST LINE) AND THE t VALUES (SECOND LINE) OF OPTIMIZATION ERRORS ON 1000-D TEST FUNCTIONS.

1000-D	CSO	CCPSO2	MLCC	sep-CMA-ES	EPUS-PSO	DMS-PSO
f_1	1.09E-21(4.20E-23) –	5.18E-13 (9.61E-14) **-2.70E+01**	8.46E-13 (5.01E-14) **-8.44E+01**	7.81E-15 (1.52E-15) **-2.57E+01**	5.53E+02 (2.86E+01) **-9.67E+01**	0.00E+00 (0.00E+00) 1.30E+02
f_2	4.15E+01(9.74E-01) –	7.82E+01 (4.25E+01) **-4.32E+00**	1.09E+02 (4.75E+00) **-6.96E+01**	3.65E+02 (9.02E+00) **-1.78E+02**	4.66E+01 (4.00E-01) **-2.42E+01**	9.15E+01 (7.14E-01) **-2.07E+02**
f_3	1.01E+03(3.02E+01) –	1.33E+03 (2.63E+02) **-6.04E+00**	1.80E+03 (1.58E+02) **-2.46E+01**	9.10E+02 (4.54E+01) 9.17E+00	8.37E+05 (1.52E+05) **-2.75E+01**	8.98E+09 (4.39E+08) **-1.02E+02**
f_4	6.89E+02(3.10E+01) –	1.99E-01 (4.06E-01) 1.11E+02	1.37E-10 (3.37E-10) 1.11E+02	5.31E+03 (2.48E+02) **-9.24E+01**	7.58E+03 (1.51E+02) **-2.24E+02**	3.84E+03 (1.71E+02) **-9.07E+01**
f_5	2.26E-16(2.18E-17) –	1.18E-03 (3.27E-03) **-1.80E+00**	4.18E-13 (2.78E-14) **-7.51E+01**	3.94E-04 (1.97E-03) -1.00E+00	5.89E+00 (3.91E-01) **-7.53E+01**	0.00E+00 (0.00E+00) 5.18E+01
f_6	1.21E-12(2.64E-14) –	1.02E-12 (1.68E-13) 5.59E+00	1.06E-12 (7.68E-14) 9.24E+00	2.15E+01 (3.19E-01) **-3.37E+02**	1.89E+01 (2.49E+00) **-3.80E+01**	7.76E+00 (8.92E-02) **-4.35E+02**
f_7	-3.83E+06(4.82E+04) –	-1.43E+04 (8.27E+01) **-3.96E+02**	-1.47E+04 (1.51E+01) **-3.96E+02**	-1.25E+04 (9.36E+01) **-3.96E+02**	-6.62E+03 (3.18E+01) **-3.97E+02**	-7.50E+03 (1.63E+01) **-3.97E+02**
$w/t/l$	–	4/1/2	5/0/2	5/1/1	7/0/0	5/0/2

have participated in the CEC 2008 competition on Large Scale Global optimization (LSGO) [43].

Based on the previous empirical analysis of the two parameters in CSO, the parameter settings used in the benchmarks are summarized in Table III. The optimization errors on 100-D, 500-D and 1000-D functions are summarized in Table IV, V and VI, respectively. In all the three tables, t values are listed together mean values and the standard deviations. A negative t value means that the statistical results of the optimization errors obtained by CSO are relatively smaller and vice versa. If the difference is statistically significant smaller, the corresponding t value is highlighted. $w/t/l$ in the last row means that CSO wins in w functions, ties in t functions, and loses in l functions.

The statistical results of the optimization errors show that CSO has significantly better overall performance in comparison with all the other five compared algorithms on 500-D, 1000-D functions. CSO and DMS-PSO have similar performance on 100-D functions, and both outperform the rest four algorithms. It seems that DMS-PSO is always able to find the global optimum of f_1 and f_5, regardless of the number of search dimensions, but has poor performance on the other five functions in comparison with CSO, especially when the dimensionality becomes higher. In comparison, MLCC has yielded the best results on f_4, which is a shifted Rastrigin function with a large number of local optima. Such outstanding

performance on f_4 should be brought about by the differential evolution variant (SaNSDE) used in MLCC.

In addition, the convergence profiles of one typical separable function (f_1) and one typical non-separable function (f_5) are plotted in Fig. 8. It can be seen that, although the convergence speed of the proposed is not so fast as the CCPSO2 or MLCC at the very beginning, it is able to perform a relatively more consistent convergence to continuously improve the solution quality.

C. Scalability to higher dimensionality

From the statistical results of the optimization errors summarized in Table IV, V and VI, it can be noticed that CSO has shown very good scalability to the search dimension, i.e., the performance does not deteriorate seriously as the dimension increases.

To further examine the search ability of CSO on the functions of even higher dimensions, e.g., 2000-D or even 5000-D, additional experiments have been performed on f_1 to f_6 of 2000 and 5000 dimensions. f_7 is excluded from this experiment for the reason that its global optimum is dimension dependent and thus it is not easy to evaluate the scalability.

It must be stressed that optimization of problems of 2000 and 5000 dimensions is very challenging for CSO since it has not adopted any particular strategies tailored for solving large scale optimization, e.g., the *divide-and-conquer* strategy. Furthermore, to the best of our knowledge, optimization of problems of a dimension larger than 1000 has only been reported by Li and Yao in [52], where 2000-dimensional f_1, f_3 and f_7 have been employed to test their proposed CCPSO2.

TABLE VII
PARAMETER SETTINGS OF CSO ON 2000-D AND 5000-D FUNCTIONS.

Parameter	Dimension	Separable	Non-separable
m	2000-D	1000	1000
	5000-D	1500	1500
φ	2000-D	0.2	0.15
	5000-D	0.2	0.15

Separable functions include f_1, f_4, f_5 and f_6. Non-separable functions include f_2 and f_3. Note that f_5 is grouped as a non-separable function because the product component becomes less significant with the increase of the dimension [52].

The parameter settings are listed in Table VII and the statistical results of optimization errors are listed in Table VIII. It can be seen that CSO continues to perform well even if the dimension is higher than 1000, especially on the three separable functions f_1, f_4 and f_6, together with f_5.

In order to get an overall picture of the scalability of CSO and the five compared algorithms, the mean optimization errors obtained by the six algorithms on all test functions of dimensions 100, 500 and 1000 are plotted in Fig 9, together with the mean optimization errors obtained by CCPSO2 and sep-CMA-ES on 2000-D f_1 and f_3 [52], as well as the mean optimization errors obtained by CSO on 2000-D and 5000-D f_1 to f_6. Unfortunately, we are not able to obtain the results of the compared algorithms on all 2000-D and 5000-D functions due to the prohibitively high time-consumption needed for optimization of such large scale problems. Nevertheless, it can be seen that CSO has shown the best scalability on f_1 and f_5, where the mean optimization errors obtained by CSO on the 2000-D and 5000-D test problems are much better than those obtained by the compared algorithms. Meanwhile, CSO shows similar scalability to CCPSO2 and MLCC on f_3 and f_6.

TABLE VIII
STATISTICAL RESULTS OF THE OPTIMIZATION ERRORS OBTAINED BY CSO ON 2000-D AND 5000-D FUNCTIONS.

	D = 2000	D = 5000
f_1	1.66E-20(3.36E-22)	1.43E-19(3.33E-21)
f_2	6.17E+01(1.31E+00)	9.82E+01(9.78E-01)
f_3	2.10E+03(5.14E+01)	7.30E+03(1.26E+02)
f_4	2.81E+03(3.69E+01)	7.80E+03(8.73E+01)
f_5	3.33E-16(0.00E+00)	4.44E-16(0.00E+00)
f_6	3.26E-12(5.43E-14)	6.86E-12(5.51E-14)

Since the time cost of one single run on a 5000-D functions is extremely expensive, the statistical results of optimization errors are averaged over 10 independent runs.

D. Influence of neighborhood control

In CSO, as introduced in Section II, there exist two versions for calculating the mean position $\bar{X}_k(t)$ in the learning strategy, one global version $\bar{X}^g_k(t)$ and one local version $\bar{X}^l_{l,k}(t)$, where the calculation of $\bar{X}^l_{l,k}(t)$ is based on a neighborhood instead of the whole swarm, refer to (6). Although the effectiveness of $\bar{X}^g_k(t)$ has already been verified by the empirical results above, it is still very interesting to investigate the influence of neighborhood control used in $\bar{X}^l_{l,k}(t)$ on the swarm diversity and thus the search performance.

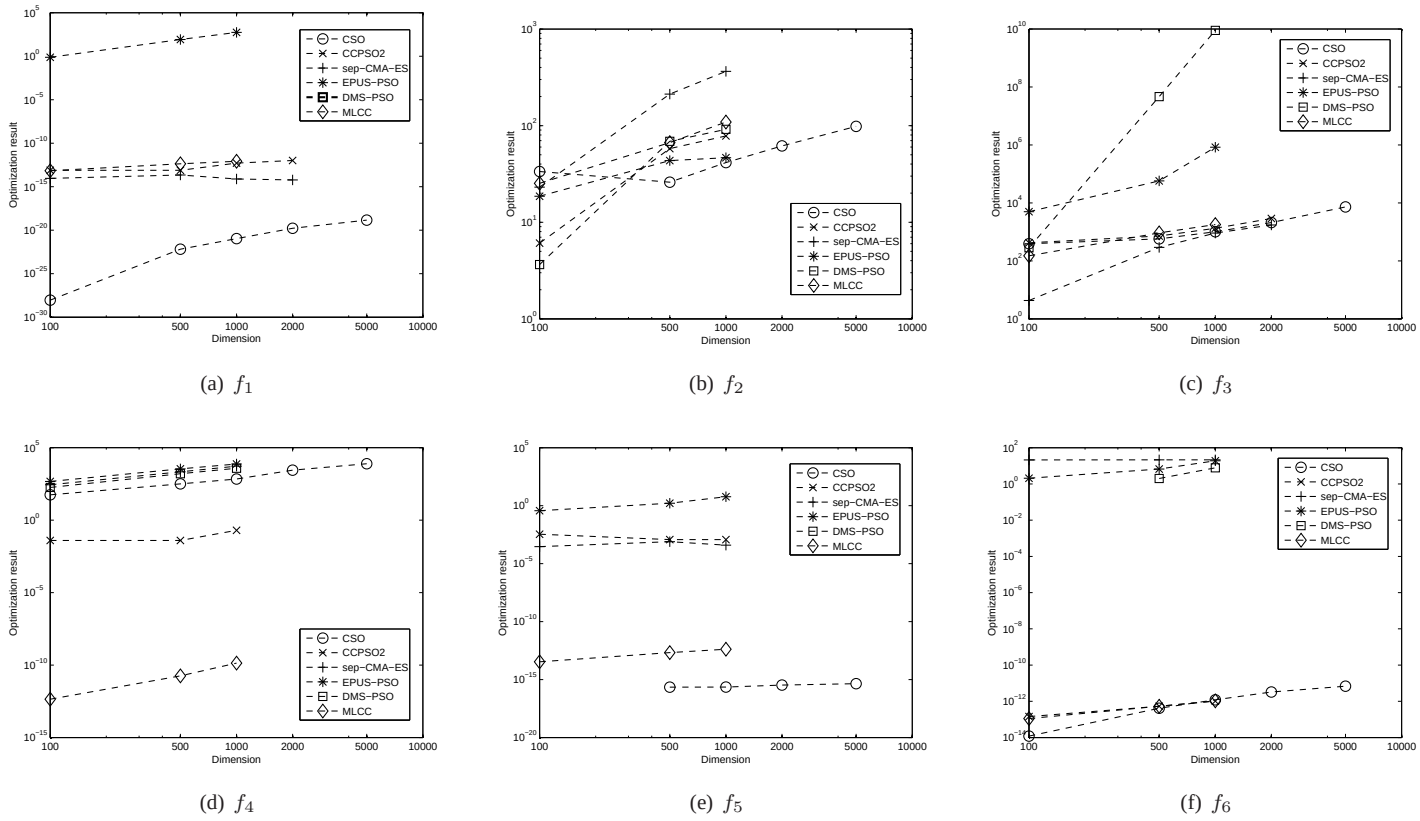
With neighborhood control, the whole swarm is dynamically divided into several neighborhoods, each neighborhood having a local mean position vector. This will enhance the swarm diversity in comparison with the original CSO where the whole swarm shares a global mean position. Intuitively, a higher degree of swarm diversity may help alleviate premature convergence but can also slow down the convergence speed to a certain extent.

For simplicity, the commonly used *ring topology* [22], [23], [59], which has been shown to be an effective neighborhood structure [60] is adopted here. In this topology, each particle takes the two immediate neighbors to form a neighborhood [61].

First, we investigate the influence of the neighborhood control on the swarm diversity. In order to obtain measurable observations, a *diversity measure* introduced in [62], [63] is adopted here to indicate the change of diversity during the search process:

$$D(P) = \frac{1}{m}\sum_{i=1}^{m}\sqrt{\sum_{j=1}^{n}(x_i^j - \bar{x}^j)^2} \quad \text{with} \quad \bar{x}^j = \frac{1}{m}\sum_{i=1}^{m}(x_i^j), \tag{27}$$

where $D(P)$ denotes the diversity of the swarm P, m is the swarm size, n is the dimension of the decision space, x_i^j is



(a) f_1 (b) f_2 (c) f_3

(d) f_4 (e) f_5 (f) f_6

Fig. 9. The statistical results of optimization errors obtained by CSO, CCPSO2, MLCC , sep-CMA-ES, EPUS-PSO and DMS-PSO on 100-D, 500-D, 1000-D f_1 to f_6 , together with the statistical results of optimization errors obtained by CCPSO2, sep-CMA-ES on 2000-D f_1, f_3 and the statistical results of optimization errors obtained by CSO on 2000-D, 5000-D f_1 to f_6. Note that due to the logarithmic scale used in the plots, errors of 0 cannot be shown.

the value of the j-th dimension of particle i, and $\bar{x}^j$ is the average value of the j-th dimension over all particles.

It can be seen from Fig. 10 that the overall swarm diversity of CSO with neighborhood control (denoted as CSO-n) is higher than that of the original CSO, which is in consistency with the expectation above.

To further assess whether the enhanced swarm diversity can have a positive influence on the search performance of CSO-n, additional numerical experiments have been conducted on 500-D and 1000-D functions. Two-tailed t-test is implemented at a significance level $\alpha = 0.05$ between the statistical results of optimization errors obtained by CSO-n and CSO. A negative t value means that the statistical results obtained by CSO-n are relatively smaller and vice versa. The smaller statistical results are highlighted.

TABLE IX
STATISTICAL RESULTS OF OPTIMIZATION ERRORS OBTAINED BY CSO-N AND CSO ON 500-D FUNCTIONS.

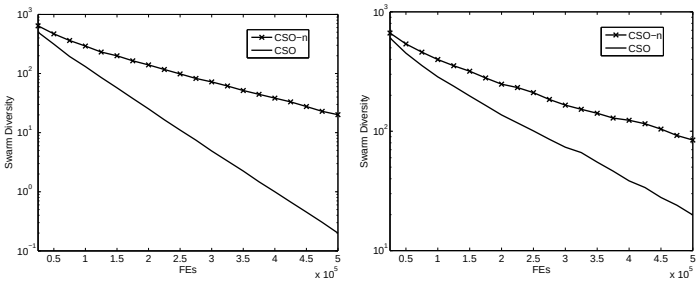
$m = 250$	CSO-n	CSO	t value
f_1	2.71E-011(5.77E-012)	**6.57E-23(3.90E-24)**	2.35E+01
f_2	4.61E+001(1.02E+000)	**2.60E+01(2.40E+00)**	3.85E+01
f_3	5.37E+002(4.00E+001)	5.74E+02(1.67E+02	-1.08E+00
f_4	3.95E+003(1.32E+002)	**3.19E+02(2.16E+01)**	1.36E+02
f_5	4.04E-012(7.00E-013)	**2.22E-16(0.00E+00)**	2.89E+01
f_6	4.90E-007(5.98E-008)	**4.13E-13(1.10E-14)**	4.10E+01
f_7	-1.68E+006(8.17E+003)	**-1.97E+06(4.08E+04)**	3.48E+01

In the first set of experiments, the same parameter settings as in the original CSO have been used for CSO-n, refer to Table III. The experimental results shown in Table IX and Table X indicate that CSO-n is outperformed by the original CSO. As discussed above, the neighborhood control is expected to generate a higher degree of swarm diversity and the experimental results in Fig. 10 has empirically confirmed this expectation. Therefore, one possible reason for the inferior performance of CSO using neighborhood control is that for these test functions, the diversity of the global CSO is already sufficient and therefore additional diversity may slow down the convergence.

TABLE X
STATISTICAL RESULTS OF OPTIMIZATION ERRORS OBTAINED BY CSO-N AND CSO ON 1000-D FUNCTIONS.

$m = 500$	CSO-n	CSO	t value
f_1	7.77E-001(2.30E-002)	**1.09E-21(4.20E-23)**	1.69E+02
f_2	8.11E+001(6.48E-001)	**4.15E+01(9.74E-01)**	1.69E+02
f_3	1.31E+007(7.74E+005)	**1.01E+03(3.02E+01)**	8.46E+01
f_4	1.02E+004(5.51E+001)	**6.89E+02(3.10E+01)**	7.52E+02
f_5	4.22E-002(1.77E-003)	**2.26E-16(2.18E-17)**	1.19E+02
f_6	5.95E-002(6.32E-003)	**1.21E-12(2.64E-14)**	4.71E+01
f_7	-2.58E+006(3.06E+004)	**-3.83E+06(4.82E+04)**	1.09E+02

In the global version of CSO, the main source of swarm diversity comes from the random pairwise competitions, where the swarm size can be an important factor to determine the amount of diversity. More specifically, a bigger swarm size is able to provide more combinations of random pairwise compe-



(a) f_1 (b) f_3

Fig. 10. The swarm diversity profiles during 500,000 Fitness Evaluations (FEs) of CSO with neighborhood control (denoted as CSO-n) and the original CSO on 500-D functions on a separable function f_1 and a non-separable function f_3 respectively.

titions, thus generating a higher degree of swarm diversity, and vice versa. Following this line of thoughts, the performance of CSO-n may be improved by reducing the swarm size. Therefore, a second set of experiments have been conducted using a different parameter setup, where the swarm size is set to $m = 150$ for 500-D functions and $m = 200$ for 1000-D functions, respectively.

TABLE XI
STATISTICAL RESULTS OF OPTIMIZATION ERRORS OBTAINED BY CSO-N AND CSO ON 500-D FUNCTIONS.

$m = 150$	CSO-n	CSO	t value
f_1	**1.51E-025(3.21E-027)**	4.10E-023(9.28E-025)	-2.20E+02
f_2	**5.23E+001(1.11E+001)**	8.20E+001(4.53E+000)	-1.24E+01
f_3	7.93E+002(1.03E+002)	9.32E+002(4.15E+002)	-1.63E+00
f_4	**4.18E+002(3.04E+001)**	6.45E+002(2.66E+001)	-2.81E+01
f_5	**3.11E-016(4.44E-017)**	2.46E-003(4.93E-003)	-2.49E+00
f_6	**4.09E-014(1.74E-015)**	1.08E+000(1.41E-001)	-3.83E+01
f_7	-1.79E+006(1.28E+004)	**-2.10E+006(5.73E+003)**	1.11E+02

TABLE XII
STATISTICAL RESULTS OF OPTIMIZATION ERRORS OBTAINED BY CSO-N AND CSO ON 1000-D FUNCTIONS.

$m = 200$	CSO-n	CSO	p value
f_1	**3.60E-018(9.38E-019)**	5.22E-013(3.70E-013)	-7.05E+00
f_2	**6.50E+001(1.10E+000)**	1.03E+002(3.24E+000)	-5.55E+01
f_3	**1.61E+003(7.96E+001)**	1.95E+003(2.08E+002)	-7.63E+00
f_4	**1.04E+003(4.85E+001)**	2.14E+003(7.51E+001)	-6.15E+01
f_5	**7.77E-016(0.00E+000)**	2.46E-003(4.93E-003)	-2.49E+00
f_6	**1.37E-010(1.84E-011)**	3.03E+000(2.67E-001)	-5.67E+01
f_7	-2.90E+006(1.69E+004)	**-4.24E+006(3.05E+004)**	1.92E+02

As shown by the statistical results of optimization errors in Table XI and Table XII, after reducing the swarm size, CSO-n is able to outperform the global CSO on most test functions studied in this work, except for f_7. Interestingly, the performance of CSO on f_7 is always better than CSO-n. As described in [43], f_7 is a very special function which has large amount of random noise and its global optimum is unknown. One possible reason for such a consequence is that f_7, as a noisy function, is very sensitive to the swarm diversity, and the original CSO, which maintains less swarm diversity, is able to achieve better performance on it.

To summarize, since the random pairwise competitions are able to generate sufficient amount of diversity in the swarm, the original CSO can work properly without using neighborhood control, if a relatively large swarm size is used for large scale optimization problems. However, use of neighborhood control, which is able to further enhance diversity, can enable us to use a smaller swarm size even for large scale problems, which is very attractive in practice.

V. CONCLUSION

In this paper, we have introduced a new swarm algorithm termed competitive swarm optimizer (CSO). The algorithm is based on a pairwise competition mechanism and adopts a novel update strategy, where neither $gbest$ nor $pbest$ is used. Theoretical proof of convergence and empirical analysis of search dynamics are given to understand the search mechanisms of CSO. Despite its simplicity in algorithmic implementation, CSO has shown to perform surprisingly well on large scale optimization problems, outperforming many state-of-the-art meta-heuristics tailored for large scale optimization. Our comparative studies conducted on 100-D, 500-D and 1000-D CEC'08 benchmark problems demonstrate that CSO performs consistently well on those test functions, especially in the high dimensional cases. The performance of CSO has been further demonstrated on 2000-D and 5000-D functions and the experimental results show that CSO has reasonably good scalability on these extremely high-dimensional problems. In addition, we have empirically investigated the influence of neighborhood control on the swarm diversity and search performance of CSO, which suggests that neighborhood control can enhance diversity and therefore enable us to user a smaller swarm size for large scale optimization problems.

In the future, we will investigate the application of CSO to solving other challenging optimization problems, such as multi-objective problems [64] and many-objective problems [65]. Application of CSO to complex real-world problems is another important future work.

ACKNOWLEDGEMENT

This work was supported in part by Honda Research Institute Europe.

REFERENCES

[1] J. Kennedy, *Swarm Intelligence*. Springer, 2006.
[2] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4. IEEE, 1995, pp. 1942–1948.
[3] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of IEEE International Conference on Evolutionary Computation*. IEEE, 1998, pp. 69–73.
[4] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of International Symposium on Micro Machine and Human Science*. IEEE, 1995, pp. 39–43.
[5] J. Kennedy, "The particle swarm: social adaptation of knowledge," in *Proceedings of IEEE International Conference on Evolutionary Computation*. IEEE, 1997, pp. 303–308.
[6] I. Montalvo, J. Izquierdo, R. Pérez, and P. L. Iglesias, "A diversity-enriched variant of discrete pso applied to the design of water distribution networks," *Engineering Optimization*, vol. 40, no. 7, pp. 655–668, 2008.
[7] A. Alfi and M.-M. Fateh, "Parameter identification based on a modified pso applied to suspension system," *Journal of Software Engineering & Applications*, vol. 3, pp. 221–229, 2010.

[8] Y.-J. Gong, J. Zhang, H. Chung, W.-n. Chen, Z.-H. Zhan, Y. Li, and Y.-h. Shi, "An efficient resource allocation scheme using particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 6, pp. 801–816, 2012.

[9] S.-Y. Ho, H.-S. Lin, W.-H. Liauh, and S.-J. Ho, "Opso: Orthogonal particle swarm optimization and its application to task assignment problems," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 38, no. 2, pp. 288–298, 2008.

[10] Z. Zhu, J. Zhou, Z. Ji, and Y.-H. Shi, "Dna sequence compression using adaptive particle swarm optimization-based memetic algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 643–658, 2011.

[11] Y. Shi *et al.*, "Particle swarm optimization: developments, applications and resources," in *Proceedings of IEEE Congress on Evolutionary Computation*, vol. 1. IEEE, 2001, pp. 81–86.

[12] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Proceedings of International Conference on Machine Learning*. Morgan Kaufmann Publishers, 1997, pp. 412–420.

[13] W.-N. Chen, J. Zhang, Y. Lin, and e. Chen, "Particle swarm optimization with an aging leader and challengers," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 2, pp. 241–258, 2013.

[14] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming VII*. Springer, 1998, pp. 591–600.

[15] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of IEEE Congress on Evolutionary Computation*. IEEE, 1999, pp. 1945–1950.

[16] ——, "Fuzzy adaptive particle swarm optimization," in *Proceedings of IEEE Congress on Evolutionary Computation*, vol. 1. IEEE, 2001, pp. 101–106.

[17] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[18] R. Cheng and M. Yao, "Particle swarm optimizer with time-varying parameters based on a novel operator," *Applied Mathematics and Information Sciences*, vol. 5, no. 2, pp. 33–38, 2011.

[19] M. Hu, T. Wu, and J. D. Weir, "An adaptive particle swarm optimization with multiple adaptive methods," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 705–720, 2013.

[20] P. N. Suganthan, "Particle swarm optimiser with neighbourhood operator," in *Proceedings of IEEE Congress on Evolutionary Computation*, vol. 3. IEEE, 1999, pp. 1958–1962.

[21] J. Kennedy, "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance," in *Proceedings of IEEE Congress on Evolutionary Computation*, vol. 3. IEEE, 1999, pp. 1931–1938.

[22] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceedings of IEEE Congress on Evolutionary Computation*, vol. 2. IEEE, 2002, pp. 1671–1676.

[23] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[24] J. J. Liang, A. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[25] B. Qu, P. Suganthan, and S. Das, "A distance-based locally informed particle swarm model for multimodal optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 387–402, 2013.

[26] J. Robinson, S. Sinton, and Y. Rahmat-Samii, "Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna," in *Proceedings of IEEE Antennas and Propagation Society International Symposium*, vol. 1. IEEE, 2002, pp. 314–317.

[27] C.-F. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 2, pp. 997–1006, 2004.

[28] N. Holden and A. A. Freitas, "A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data," in *Proceedings fo the IEEE Swarm Intelligence Symposium*. IEEE, 2005, pp. 100–107.

[29] P. Shelokar, P. Siarry, V. K. Jayaraman, and B. D. Kulkarni, "Particle swarm and ant colony algorithms hybridized for improved continuous optimization," *Applied mathematics and computation*, vol. 188, no. 1, pp. 129–142, 2007.

[30] J. Liang and P. Suganthan, "Dynamic multi-swarm particle swarm optimizer," in *Proceedings of IEEE Swarm Intelligence Symposium*. IEEE, 2005, pp. 124–129.

[31] Z.-H. Zhan, J. Zhang, Y. Li, and H.-H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 6, pp. 1362–1381, 2009.

[32] R. Brits, A. P. Engelbrecht, and F. van den Bergh, "Locating multiple optima using particle swarm optimization," *Applied Mathematics and Computation*, vol. 189, no. 2, pp. 1859–1883, 2007.

[33] M. Daneshyari and G. G. Yen, "Cultural-based multiobjective particle swarm optimization," *IEEE Transactions on Cybernetics*, vol. 41, no. 2, pp. 553–567, 2011.

[34] N. Higashi and H. Iba, "Particle swarm optimization with gaussian mutation," in *Proceedings of IEEE Swarm Intelligence Symposium*. IEEE, 2003, pp. 72–79.

[35] B. Liu, L. Wang, Y.-H. Jin, F. Tang, and D.-X. Huang, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons & Fractals*, vol. 25, no. 5, pp. 1261–1271, 2005.

[36] Z.-H. Zhan, J. Zhang, Y. Li, and Y.-H. Shi, "Orthogonal learning particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 6, pp. 832–847, 2011.

[37] H. Gao and W. Xu, "A new particle swarm algorithm and its globally convergent modifications," *IEEE Transactions on Cybernetics*, vol. 41, no. 5, pp. 1334–1351, 2011.

[38] Y. V. Pehlivanoglu, "A new particle swarm optimization method enhanced with a periodic mutation strategy and neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 436–452, 2013.

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

[40] S. Baskar and P. N. Suganthan, "A novel concurrent particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 1. IEEE, 2004, pp. 792–796.

[41] G. G. Yen and M. Daneshyari, "Diversity-based information exchange among multiple swarms in particle swarm optimization," *International Journal of Computational Intelligence and Applications*, vol. 7, no. 01, pp. 57–75, 2008.

[42] R. Cheng, C. Sun, and Y. Jin, "A multi-swarm evolutionary framework based on a feedback mechanism," in *Proceedings of IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 718–724.

[43] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y.-P. Chen, C.-M. Chen, and Z. Yang, "Benchmark functions for the cec'2008 special session and competition on large scale global optimization," *Nature Inspired Computation and Applications Laboratory, USTC, China*, 2007.

[44] Y. Jin and B. Sendhoff, "Fitness approximation in evolutionary computation-A survey," in *Genetic and Evolutionary Computation Conference (GECCO)*, 2002, pp. 1105–1112.

[45] D. Lim, Y. Jin, Y.-S. Ong, and B. Sendhoff, "Generalizing surrogate-assisted evolutionary computation," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 329–355, 2010.

[46] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theoretical computer science*, vol. 344, no. 2, pp. 243–278, 2005.

[47] D. Karaboga and B. Akay, "A survey: algorithms simulating bee swarm intelligence," *Artificial Intelligence Review*, vol. 31, no. 1-4, pp. 61–85, 2009.

[48] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[49] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[50] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letters*, vol. 85, no. 6, pp. 317–325, 2003.

[51] J. L. Fernandez-Martinez and E. Garcia-Gonzalo, "Stochastic stability analysis of the linear continuous and discrete pso models," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 3, pp. 405–423, 2011.

[52] X. Li and Y. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 2, pp. 1–15, 2011.

[53] Z. Yang, K. Tang, and X. Yao, "Multilevel cooperative coevolution for large scale optimization," in *Proceedings of IEEE Congress on Evolutionary Computation*. IEEE, 2008, pp. 1663–1670.

[54] R. Ros and N. Hansen, "A simple modification in cma-es achieving linear time and space complexity," *Parallel Problem Solving from Nature–PPSN X*, pp. 296–305, 2008.
[55] S.-T. Hsieh, T.-Y. Sun, C.-C. Liu, and S.-J. Tsai, "Solving large scale global optimization using improved particle swarm optimizer," in *Proceedings of IEEE Congress on Evolutionary Computation*. IEEE, 2008, pp. 1777–1784.
[56] M. Potter and K. De Jong, "A cooperative coevolutionary approach to function optimization," *Parallel Problem Solving from Nature, PPSN III*, pp. 249–257, 1994.
[57] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences*, vol. 178, no. 15, pp. 2985–2999, 2008.
[58] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.
[59] M. A. Montes de Oca, T. Stutzle, M. Birattari, and M. Dorigo, "Frankenstein's pso: a composite particle swarm optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1120–1132, 2009.
[60] X. Li, "Niching without niching parameters: particle swarm optimization using a ring topology," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 1, pp. 150–169, 2010.
[61] X. Li and K. Deb, "Comparing lbest pso niching algorithms using different position update rules," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–8.
[62] O. Olorunda and A. Engelbrecht, "Measuring exploration/exploitation in particle swarms using swarm diversity," in *Proceedings of IEEE Congress on Evolutionary Computation*. IEEE, 2008, pp. 1128–1134.
[63] A. Ismail and A. Engelbrecht, "Measuring diversity in the cooperative particle swarm optimizer," *Swarm Intelligence*, vol. 7461, pp. 97–108, 2012.
[64] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons Hoboken, NJ, 2001.
[65] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: A short review," in *Proceedings of IEEE Congress on Evolutionary Computation*. IEEE, 2008, pp. 2419–2426.