Information Sciences 329 (2016) 52–72

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Accelerating particle swarm optimization using crisscross search

Anbo Meng*, Zhuan Li, Hao Yin, Sizhe Chen, Zhuangzhi Guo

Faculty of Automation, Guangdong University of Technology, Guangzhou 510006, China

ARTICLE INFO

Article history:
Received 6 October 2014
Revised 4 August 2015
Accepted 15 August 2015
Available online 11 September 2015

Keywords:
Particle swarm optimization (PSO)
Global search
Crisscross search optimization (CSO)
Horizontal crossover
Vertical crossover

ABSTRACT

Although the particle swarm optimization (PSO) algorithm has been widely used to solve many real world problems, it is likely to suffer entrapment in local optima when addressing multimodal optimization problems. In this paper, we report a novel hybrid optimization algorithm called crisscross search particle swarm optimization (CSPSO), which is different from PSO and its variants in that each particle is directly represented by *pbest*. The population of particles in CSPSO is updated by modified PSO as well as crisscross search optimization (CSO) in sequence within each iteration. CSO is incorporated as an evolutionary catalytic agent that has powerful capability of searching for *pbests* of high quality, therefore accelerating the global convergence of PSO. CSO enhances PSO by two search operators, namely horizontal crossover and vertical crossover. The horizontal crossover further enhances PSO's global convergence ability while the vertical crossover can enhance swarm diversity. Several benchmark functions are used to test the feasibility and efficiency of the proposed algorithm. The experimental results show that CSPSO outperforms other state-of-the-art PSO variants significantly in terms of global search ability and convergence speed on almost all of the benchmark functions tested.

1. Introduction

Inspired by observing the natural swarming behavior of fish schooling and bird flocking, a popular stochastic optimization technique called particle swarm optimization (PSO) was proposed in [20,35,56]. Unlike other population-based evolutionary algorithms, PSO explores the search space according to two important "leaders": *pbest* and *gbest*, which are the previous best positions achieved by the individual particle and the swarm so far, respectively. As the PSO algorithm is simple in concept, easy to implement and computationally inexpensive, it attracts the attention of many scholars and researchers in the last two decades. PSO has now been successfully applied to a wide range of application areas such as electric power systems [2,5,8,10,25,29,51,53,70], engineering design [27,30,32,67,76,81], neural networks [1,6,16,24,43,60,65,71,74] and so on.

Although PSO has a fast convergence rate and global search ability when searching in the unimodal problem space, it often gets stuck and trapped in local minima when applied to more complex problems such as multimodal or shifted and rotated functions [9]. So far, studying how to enhance PSO's swarm diversity and further improve its global search ability has become a focus subject.

In this paper, we introduce a novel crisscross search particle swarm optimizer (CSPSO), which exhibits a competitive advantage over other improved PSO versions in terms of global search ability and convergence speed on almost all of the benchmark functions tested in this paper. The CSPSO algorithm accelerates the search process toward converging to the global optimum by

* Corresponding author. Tel.: +86 1392 8799 469; fax: +86 02039322563.
E-mail address: menganbo@vip.sina.com (A. Meng).

http://dx.doi.org/10.1016/j.ins.2015.08.018
0020-0255/

incorporating a recently developed crisscross search optimization (CSO) inspired by the Confucian doctrine of the golden mean and the crossover operation in genetic algorithms [47]. CSO adopts a duo search mechanism including horizontal crossover and vertical crossover, which reproduce a population of moderation solutions at each generation by performing different crossover operations in opposite directions. Only those moderation solutions, which outperform their counterparts in the parent population, survive in the next generation. Otherwise, they are eliminated by their parent competitors (i.e., *pbests*). Considering that CSO just maintains a population of *pbests* of high quality, we attempt to directly use *pbest* to represent a candidate solution in CSPSO. Accordingly, the velocity expression of a particle needs to be readjusted, which will be discussed in Section 3. To validate the performance of the CSPSO algorithm, well-known benchmark functions are used to test its solution accuracy and convergence speed. The different experimental results show that the proposed CSPSO algorithm significantly outperforms most of the other state-of-the-art PSO versions included in this paper in terms of global search ability and convergence speed.

The rest of this paper is organized as follows. Some related work on PSO is reviewed in Section 2. Section 3 introduces CSPSO, in which the procedures of the algorithm are presented in detail. In Section 4, the effect of crisscross search on CSPSO is discussed and different experimental tests are used to verify the performance of the proposed algorithm. Finally, conclusion and future work are included in Section 5.

2. Related work on particle swarm optimization

2.1. Original PSO

PSO is a population-based optimization algorithm. It is known that PSO maintains two populations: a population of particles' current positions and a population of particles' best positions (i.e., *pbests*) achieved so far. The former is regarded as the candidate solutions in the problem space and the latter is used to guide the former's updating. In PSO, each particle is associated with two properties (e.g., a velocity vector V and a position vector X). In a D-dimensional space, suppose the velocity vector and the position vector of particle i ($i = 1, 2, \ldots, M$) are represented by $V_i = (v_i^1, v_i^2, \cdots, v_i^D)$ and $X_i = (x_i^1, x_i^2, \cdots x_i^D)$, they are updated at each iteration according to Eqs. (1) and (2):

$$v_i^d \leftarrow v_i^d + c_1 \times r_1 \times (pbest_i^d - x_i^d) + c_2 \times r_2 \times (gbest^d - x_i^d) \tag{1}$$

$$x_i^d \leftarrow x_i^d + v_i^d \tag{2}$$

where c_1 and c_2 are positive acceleration constants representing the weighting of stochastic acceleration terms that pull each particle toward *pbest* and *gbest* positions; r_1 and r_2 are two independently uniformly distributed random variables in the range [0,1]; x_i denotes the position of ith particle; $V_i = (v_i^1, v_i^2, \cdots, v_i^D)$ represents the velocity change for particle i; $pbest_i = (pbest_i^1, pbest_i^2, \cdots, pbest_i^D)$ is the best previous position achieved so far by particle i and $gbest = (gbest^1, gbest^2, \cdots, gbest^D)$ is the current best position discovered by the whole swarm.

PSO works in a very simple way. After its particles are initialized with a population of random positions in the problem space, PSO begin to enter a loop in order to continue to search for optimal solutions by updating the particles' velocities and positions according to (1) and (2). Since the trajectory of each particle in the search space follows its *pbest* and *gbest* throughout the iterative process, PSO often exhibits fast-converging speed in the process of optimization. However, PSO also easily suffers the premature convergence problem. According to Angeline [5], although PSO converges to an optimum much faster than other evolutionary algorithms, it usually cannot improve the quality of the solutions at the last period of iterations especially when optimizing high dimensional multi-modal problems. To enhance PSO's performance, various improved PSO versions are developed in recent years. These improvements on PSO can be roughly classified into three categories including modifying the parameters, adopting different neighborhood topologies, hybridizing PSO with other auxiliary search techniques, which are discussed in the next section.

2.2. Improved PSO versions

2.2.1. Modifying the parameters

In order to better control the scope of the search, Shi and Eberhart [57] proposed a new parameter w called inertia weight for balancing the global and local search. Eq. (1) is changed to:

$$v_i^d \leftarrow w \times v_i^d + c_1 \times r_1 \times (pbest_i^d - x_i^d) + c_2 \times r_2 \times (gbest^d - x_i^d). \tag{3}$$

Initially the inertia weight was kept static as 0.4 during the search process. Later, Shi and Eberhart [58] conducted the empirical study on the linearly decreasing inertia weight from 0.9 to 0.4, but the results illustrate that such adjustment way may make PSO lose global search ability at the end of a run. To overcome such issue, a self-adaptive strategy for adjusting the inertia weights is employed in Shi and Eberhart [59].

Clerc and Kennedy [15] introduced a constriction factor K in order to insure the convergence of PSO and Eq. (1) is changed to:

$$v_i^d \leftarrow K \times [v_i^d + c_1 \times r_1 \times (pbest_i^d - x_i^d) + c_2 \times r_2 \times (gbest^d - x_i^d)] \tag{4}$$

$$K = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|}, \quad c_1 + c_2 = \phi > 4.0 \tag{5}$$

Typically, when Clerc's constriction method is used, $\phi = c_1 + c_2 > 4$, and K is a function of c_1 and c_2. Usually, ϕ is set to 4.1 ($c_1 = c_2 = 2.05$), and the constant multiplier K is thus approximately 0.729.

In recent years, several other PSO variants adopt the parameters adjusting strategy to enhance PSO's performance. For example, Zhou et al. [80] applied random position in conventional PSO with linearly decreasing inertia weight (RPPSO). Zhan et al. [77] introduced an adaptive particle swarm optimization (APSO). In their research, the optimization strategy enables the automatic control of inertia weight, acceleration coefficients, and other algorithmic parameters at runtime to improve the search efficiency and convergence speed. Besides, by using the elitist learning mechanism, the globally optima could jump out of the likely local optima. Tang et al. [63] proposed another improved PSO version called the feedback learning swarm optimization algorithm with quadratic inertia weight (FLPSO-QIW). In FLPSO-QIW, the inertia weight is calculated by a designed quadratic function, and the feedback fitness information of each particle is used to automatically design the learning probabilities. Ratnaweera et al. [52] proposed a self-organizing hierarchical particle optimizer with time-varying acceleration coefficients (HPSO-TVAC). By observing the distribution of samples, Kennedy [34] proposed the bare bones particle swarm optimization algorithm (BBPSO). In BBPSO, the interaction probability *IP* is set to 0.5. Approximately half the time, the vector element $x_i^d = pbest_i^d$ will be tested. The other half time, the vector element x_i^d will be derived from a Gaussian distribution with mean $x_i^d = \frac{pbest_i^d + gbest^d}{2}$ on each dimension d and standard deviation $s.d.(x_i^d) = |pbest_i^d - gbest^d|$. Inspired by the effect of aging on the cultural diversity of a social animal colony, Chen et al. [12] proposed a PSO with an aging leader and challengers (ALC-PSO), which is characterized by assigning the leader of the population with a growing age and a lifespan, and allowing the other individuals to challenge the leadership when the leader becomes aged.

By adjusting parameters according to certain rules, although PSO can improve the convergent performance, the way of assigning the parameter settings to PSO is often problem-dependent. In general, no single parameter setting exists which can be applied to all problems and performs dominantly better than other parameter settings [23].

2.2.2. Adopting different topologies for PSO

To increase the diversity of the swarm, PSO versions based on various topologies have also been developed in recent years. In the local versions of PSO, the velocity update rule (3) is changed to

$$v_i^d \leftarrow w \times v_i^d + c_1 \times r_1 \times (pbest_i^d - x_i^d) + c_2 \times r_2 \times (lbest_i^d - x_i^d) \tag{6}$$

where $lbest_i$ is the current best position in the neighborhood topology of particle i.

The salient neighborhood structures applied to PSO include the ring topology [19], the von Neumann topology [37] and the small world topology [33]. Kennedy and Mendes [36] have shown that a larger neighborhood is suitable for addressing unimodal problems while a smaller neighborhood could work better on multimodal problems. To overcome the shortcoming of the fixed neighborhoods, some dynamical neighborhood structures have been proposed by Suganthan [61], Liang and Suganthan [40], and Hu and Eberhart [28]. Besides, Liang et al. [39] proposed a comprehensive learning particles swarm optimization (CLPSO) for global optimization of multimodal functions. In CLPSO, particles are allowed to learn from different lbest positions on different dimensions. Another typical PSO version proposed by Mendes et al. [46] is called the fully informed particle swarm (FIPS) algorithm. In FIPS, all members in the neighborhood can offer their search information fairly and the velocity adjustment is not only influenced by the best position in the particle's neighborhood, but also by positions in other neighborhoods. The other PSO variants such as TSPSO [80], RDNEMPSO [48], PS2O [7] are also developed based on the neighborhood concept.

Compared with the original PSO, the topology structure based PSO versions have more powerful ability in terms of keeping swarm diversity and preventing premature convergence. However, because all the particles in the swarm update the velocities and positions in their respective neighborhood topologies, the computational cost will be greatly increased undoubtedly.

2.2.3. Hybridizing PSO with other auxiliary search techniques

Researchers have also hybridized PSO with other auxiliary search techniques. A natural evolution of the population based search algorithms like that of PSO can be achieved by integrating the methods that have already been tested successfully for solving complex problems [64]. The combination of PSO with other evolutionary computation techniques like selection [3], mutation [4] and crossover [44] of GA has now become a popular technique for enhancing PSO performance. Because PSO and evolutionary algorithms (EAs) including GA are both based on population, such hybridization is readily formulated. Angeline [5] applied a tournament selection process that replaced the velocities and positions of poorly performing particles with those of better performing particles. Engelbrecht [22] introduced a heterogeneous PSO (HPSO). In HPSO, particles are allowed to follow different search behaviors in terms of the particle position and velocity updates selected from a behavior pool. Two versions of the HPSO were proposed, the one static, where the behaviors do not change and a dynamic version where a particle may change its behavior during the search process if it cannot improve its personal best position. Nepomuceno et al. [50] proposed a self-adaptive heterogeneous PSO and it gains a better performance than the original HPSO on the CEC 2013 benchmark functions. Aiming at the problem that HPSO lacks the capacity to produce multiple minimum, Higashi and Iba [26] introduced Gaussian mutation to PSO, which improves the search efficiency by employing a special search strategy at the middle and final stages by gradually reducing the appearance ratio of Gaussian mutation at the initial stage. In the catfish PSO [13], when the *gbest* value

does not change after several iterations, the 10% of particles with the worst fitness value are removed, and the equal number of catfish particles are randomly positioned at extreme points of the search space. Also, chaotic catfish PSO is proposed by Chuang et al. [14] to increase the search capability of catfish PSO. Liu et al. introduced center particle swarm optimization (Center PSO) [42] to improve the performance. Xi et al. [71] introduced an improved quantum-behaved PSO, which introduced a weight parameter into the calculation of the mean best position in QPSO in order to render the importance of particles in the swarm when they are evolving. Montes de Oca et al. [49] designed a new composite PSO algorithm called Frankenstein's PSO (FPSO), which combines a number of algorithmic components that showed distinct advantages in optimization speed and reliability. Hong et al. [41] proposed a two-layer PSO with intelligent division of labor (TLPSO-IDL), in which the IDL module is developed in the latter to adaptively divide the swarm into the exploration and exploitation sections. Also, researchers have enhanced the performance of PSO by incorporating GA [28,31,38,45,64,73] and ACO [21,55,72]. The other hybrid techniques with PSO can be referred to [23].

3. The proposed CSPSO algorithm

The CSPSO algorithm differs from the PSO algorithm in that the candidate solutions are directly represented by particles' *pbest*s. As a result, the CSPSO algorithm always maintains a population of *pbest*s, which is updated by the modified PSO search as well as the crisscross search in sequence from generation to generation. As the crisscross operation includes the horizontal crossover search and the vertical crossover search, thus, there exist three search phases that can promote each other so as to find as many particles' *pbest*s of high quality as possible.

3.1. Modified PSO

In the modified PSO search phase, the PSO renewal expressions (1) and (2) are rewritten with

$$v_i^d \leftarrow w \times v_i^d + c_1 \times r_1 \times (mbest^d - pbest_i^d) + c_2 \times r_2 \times (gbest^d - pbest_i^d) \tag{7}$$

$$X_i^d \leftarrow pbest_i^d + v_i^d \tag{8}$$

where $mbest^d$ ($d = 1, 2, \ldots D$) is the average value of all individuals, $X_i{}^d$ is the newly generated position of $pbest_i{}^d$, r_1 and r_2 are two random variables in the range [0,1]. From Eqs. (7) and (8), it can be observed that the modified PSO differentiates from the original PSO in two aspects. First, since the candidate solutions are directly represented by *pbest*s, we update the velocity $v_i{}^d$ of the dth dimension of the ith particle's $pbest_i$ by using $mbest^d$ and $gbest^d$. Second, the updated position X_i cannot directly enter the next generation. Only its fitness value is better than that of its parent $pbest_i$ achieved by the crisscross search except for first generation, it can survive and then replaces $pbest_i$, else it will be eliminated. For the sake of convenience, the symbol © is used to stand for the competition operator. The $pbest_i$ updating of the ith particle' is then expressed as Eq. (9).

$$pbest_i \leftarrow X_i © pbest_i \tag{9}$$

In this paper, the competitive operation between the offspring population and its parent population is run through subsequent crisscross search.

3.2. CSO

In CSPSO, CSO can be regarded as a catalytic agent that has powerful ability to search for the *pbest*s of high quality so as to accelerate the global convergence of PSO without sacrificing its swarm diversity. Such ability benefits much from the moderation solutions generated by horizontal crossover search as well as vertical crossover search from generation to generation. Here horizontal crossover samples the moderation solutions in a half population of separate hypercubes with a larger probability, but on its periphery with a smaller probability. The vertical crossover generates the moderation solution by performing an arithmetic crossover operation on a few whole dimensions of all the particles, which has powerful ability to facilitate the stagnant dimensions to escape out of the local minima so as to keep the swarm diversity [47].

3.2.1. Horizontal crossover search

The horizontal crossover search generates the moderation solutions by executing a special arithmetic crossover operation on all the dimensions between two different particles in the population. Suppose the ith parent particle $pbest_i$ and the jth parent particle $pbest_j$ are chosen to carry out the horizontal crossover operation at the dth dimension, their dimension offspring can be reproduced as below:

$$MH_i^d = r_1^d \cdot pbest_i^d + (1 - r_1^d) \cdot pbest_j^d + c_1^d \cdot (pbest_i^d - pbest_j^d) \tag{10}$$

$$MH_j^d = r_2^d \cdot pbest_j^d + (1 - r_2^d) \cdot pbest_i^d + c_2^d \cdot (pbest_j^d - pbest_i^d) \tag{11}$$

where r_1^d and r_2^d are two random variables uniformly distributed in the range [0,1]. c_1^d and c_2^d are the expansion coefficients uniformly distributed in the range [−1,1]. Moderation solutions are expressed as MH_i and MH_j, which are the offspring of $pbest_i$ and $pbest_j$, respectively.

According to Eqs. (10) and (11), it can be seen that the generated moderation solutions consist of two parts: the first component comes from the arithmetic crossover widely applied in the real value crossover of genetic algorithm, which makes the reproduced offspring are randomly located within the scope of two parent particles. The second part introduced an expansive coefficient c that is used to enhance the global search ability. According to the probability density of the moderation solutions in D-dimension space in Meng et al. [47], the horizontal crossover operator reproduces the offspring (i.e. moderation solutions) in different hypercubes with larger probability while in the periphery of each hypercube with a decreasing probability.

After all the moderation solutions are generated, they need to compete with their parent *pbest*s stemming from the PSO search. Only those with better fitness can survive and the newly updated population of *pbest*s will serve as the parent population of the vertical crossover search.

3.2.2. Vertical crossover search

The vertical crossover plays an important role in the maintenance of swarm diversity because it is able to prevent some dimensions from suffering stagnancy. Every time the vertical crossover is implemented by performing the arithmetic crossover operation on all the particles between two different dimensions. Suppose the d_1th and d_2th dimensions of all the particles (represented by the population of *pbest*s stemming from the horizontal crossover search) are used to carry out the vertical crossover operation, the d_1th dimension offspring of particle i can be expressed as below:

$$MV_i^d = r \cdot pbest_i^{d_1} + (1 - r) \cdot pbest_i^{d_2} \tag{12}$$

where r is a uniformly distributed random value in the range [0,1].

According to Eq. (12), we observe that all the particles reproduce their moderation solutions by adopting the same single parent reproduction strategy. Such mechanism can prevent against the stagnancy of certain dimensions in the evolutionary process, which often occurs on PSO when addressing multimodal problems, especially when the particles' *pbests* are adopted as the potential solutions. For the vertical crossover operator, it needs to do the normalization operation on the population of parent particles according to the upper and lower limits of each dimension to ensure that the dimension offspring reproduced by the vertical crossover can locate within the boundary of each dimension. Compared to the horizontal crossover, another interesting difference is that the vertical crossover operator uses the crossover probability P_v to control how many dimensions in the population participate in the arithmetical crossover operation. According to the suggestion in Meng et al. [47], the crossover probability P_v is suggested to be set in the range [0.4,0.8] because that about 20–40% of the dimensions in the swarm may be simultaneously trapped into stagnancy when optimizing the multimodal functions. When the vertical crossover operation is finished, it also needs to perform the reverse normalization operation on *MV*. Similar to the first two search phases, the newly generated moderation solutions *MV* need to compete with their parent population *pbest*s stemming from horizontal crossover search. Only those whose fitness value is better than its parent competitor can survive and replace its parent *pbest*. After the vertical crossover search, the newly updated population of *pbest*s will serve as the parent population of the modified PSO search.

All in all, the three search phases described above are promoted mutually. In the horizontal crossover search phase, the population of *pbest*s updated by modified PSO search must compete with the population of moderation solutions reproduced by the horizontal crossover search. In the vertical crossover search phase, the population of *pbest*s updated by horizontal search must compete with the population of moderation solutions reproduced by the vertical crossover search. We observed that such competitive mechanism makes the particles move rapidly to the search region with better fitness and quicken the convergence rate to the global optimum. This global search ability of the proposed CSPSO algorithm owes much to the combination with the novel crisscross search in CSO that has powerful capability of searching for *pbest*s of high quality.

3.3. Procedure of the proposed CSPSO algorithm

A detailed step-by-step procedure for the proposed CSPSO algorithm is described as follows:

Step 1 *Initialization.* The initial population is represented as a swarm of $pbest_i$ ($i = 1, 2, \ldots, M$), which are initialized with random positions within the D-dimensional search space. The control parameters are set as follows: the initial velocity $V = 0$, the acceleration constants $c_1 = c_2 = 2$, the inertia weight $w = 0.4$, and the vertical crossover probability $p_v = 0.8$.

Step 2 *Modified PSO search.* In this search phase, for every particle i, the velocity of $pbest_i$ is adjusted by Eq. (7), the new generated position X_i is achieved by Eq. (8), if X_i outperforms $pbest_i$, then X_i becomes the new $pbest_i$, else $pbest_i$ keeps unchanged.

Step 3 *Horizontal crossover search.* In this search phase, all the particles in the population are randomly divided into $M/2$ pairs without repetition, then each pair of particles reproduces a couple of moderation solutions according to Eqs. (10) and (11). At the end of this search phase, the population of *pbest*s stemming from the modified PSO search is updated through competition with the population of moderation solutions generated by the horizontal crossover search.

Step 4 *Vertical crossover search*. In this search phase, all the dimensions in the population are randomly divided into $D/2$ pairs without repetition. However, according to the vertical crossover probability, only a few paired dimensions have the chance to perform the arithmetic crossover operation, which just reproduces a single dimension offspring so as to facilitate the stagnant dimension to jump out of the local optimum without destroying another dimension. To avoid the offspring out of the limited dimension range, it needs to do the normalization and reverse normalization operation on parent population and offspring population, respectively. At the end of this search phase, the population of *pbest*s stemming from the horizontal crossover search is updated through competition with the population of moderation solutions generated by the vertical crossover search.

Table 1
Benchmark functions (*note*: o is the shifted vector and M is the transform matrix, which can be referred to Suganthan et al. [[62]]).

Test functions		D	Search range	Initialization range	Global opt. $x*$	$f_{\min}$	Name
Unimodal	$f_1(x)=\sum_{i=1}^{n} x_i^2$	30	$[-100,100]^D$	$[-100,50]^D$	$\{0\}^D$	0	Sphere
	$f_2(x)=\sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	30	$[-10,10]^D$	$[-10,5]^D$	$\{0\}^D$	0	Schwefel's P2.22
	$f_3(x)=\sum_{i=1}^{n-1} [100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	30	$[-10,10]^D$	$[-10,10]^D$	$\{1\}^D$	0	Rosenbrock's
	$f_4(x)=\sum_{d=1}^{D} (\sum_{j=1}^{d} x_{i,j})^2$	30	$[-100,100]^D$	$[-100,50]^D$	$\{0\}^D$	0	Schwefel 1.2
Multimodal	$f_5(x)=10n+\sum_{i=1}^{n} (x_i^2-10\cos(2\pi x_i))$	30	$[-5.12,5.12]^D$	$[-5.12,2]^D$	$\{0\}^D$	0	Rastrigin
	$f_6(x)=\sum_{i=1}^{n} (z_i^2-10\cos(2\pi z_i)+10)$	30	$[-5.12,5.12]^D$	$[-5.12,2]^D$	$\{0\}^D$	0	Noncontinuous Rastrigin
	$f_7(x)=-20\exp(-0.2\sqrt{1/n\sum_{i=1}^{n} x_i^2}) - \exp(1/n\sum_{i=1}^{n}\cos(2\pi x_i))+20+e$	30	$[-32,32]^D$	$[-32,20]^D$	$\{0\}^D$	0	Ackley
	$f_8(x)=\frac{1}{4000}\sum_{i=1}^{n} x_i^2-\prod_{i=1}^{n}\cos(x_i/\sqrt{i})+1$	30	$[-600,600]^D$	$[-600,200]^D$	$\{0\}^D$	0	Griewank
	$f_9(x)=\sum_{i=1}^{n} -x\sin(\sqrt{x_i})$	30	$[-500,500]^D$	$[-500,500]^D$	$\{420.96\}^D$	−12569.48	Schwefel's
Shifted and Rotated	$f_{10}(x)=\sum_{i=1}^{n-1} [100(z_{i+1}-z_i^2)^2+(z_i-1)^2]+390$ $z=x-o+1$	30	$[-100,100]^D$	$[-100,100]^D$	o	390	Shifted Rosenbrock's
	$f_{11}(x)=\sum_{i=1}^{n} (z_i^2-10\cos(2\pi z_i)+10)-330, z=x\times M$	30	$[-5,5]^D$	$[-5,2]^D$	$\{0\}^D$	−330	Rotated Rastrigin's
	$f_{12}(x)=\sum_{i=1}^{n} (10^6)^{\frac{i-1}{n-1}} z_i^2-450, z=(x-o)\times M$	30	$[-100,100]^D$	$[-100,100]^D$	o	−450	Shifted Rotated high conditioned elliptic
	$f_{13}(x)=\sum_{i=1}^{n} (z_i^2-10\cos(2\pi z_i)+10)-330$ $z=(x-o)\times M$	30	$[-5,5]^D$	$[-5,5]^D$	o	−330	Shifted Rotated Rastrigin's
	$f_{14}(x)=\frac{1}{4000}\sum_{i=1}^{n} z_i^2-\prod_{i=1}^{n}\cos(z_i/\sqrt{i})+1$ $z=x-o$	30	$[-600,600]^D$	$[-600,600]^D$	o	0	Shifted Griewank
	$f_{14}(x)=\frac{1}{4000}\sum_{i=1}^{n} z_i^2-\prod_{i=1}^{n}\cos(z_i/\sqrt{i})+1$ $z=x\times M$	30	$[-600,600]^D$	$[-600,200]^D$	$\{0\}^D$	0	Rotated Griewank
	$f_{16}(x)=-20\exp(-0.2\sqrt{1/n\sum_{i=1}^{n} z_i^2}) - \exp(1/n\sum_{i=1}^{n}\cos(2\pi z_i))+20+e, z=(x-o)\times M$	30	$[-32,32]^D$	$[-32,32]^D$	o	0	Shifted Rotated Ackley
	$f_{17}(x)=\sum_{i=1}^{n} (\sum_{j=1}^{i} z_j)^2-450, z=x-o$	30	$[-100,100]^D$	$[-100,100]^D$	o	−450	Shifted Schwefel 1.2

Step 5 *Terminal condition.* The maximum number of function evaluation (*Max*FEs) is used as the terminal condition. If the number of function evaluation (FEs) is larger than *Max*FEs, the algorithm terminates. Otherwise, go to Step 2 for a new round of iteration.

4. Numerical results

In this section, five experimental tests are used to investigate the performance of the proposed CSPSO algorithm from different aspects based on several well-known benchmark functions. The first test is used to measure the effects of horizontal crossover and vertical crossover on CSPSO, respectively. The second test is designed to investigate how much the improvement of the results depends on the crisscross search and on the choice of using only one population. The third test is used to validate the comprehensive performance of the proposed algorithm by comparison with other state-of-the-art PSO variants. The last two tests aim to further exhibit the excellent performance of the proposed algorithm. We done a comparative study by using two popular PSO variants in the fourth test and we also tested a set of 10 CEC 2005 shifted benchmark functions in the fifth experiment.

In the last four experimental tests, we used a two-stage method (i.e., the statistical Friedman test and then a post-hoc test) to compare the results of different algorithms across multiple datasets (i.e., the benchmark problems tested) according to the suggestions in [17,18]. The Friedman test is a nonparametric analogue of the parametric two-way analysis of variance. For the Friedman test, it needs the computation of the average ranks of algorithms. Once the statistics [18] have been computed, it is possible to compute a p-value through normal approximations [1]. If the existence of significant differences is found (that is, the null hypothesis is rejected), a post-hoc procedure will be proceed to compare the performance differences between CSPSO and the other PSO variants. In this work, for pairwise comparisons, we also reported the adjusted p-values (APVs) achieved by two post hoc test procedures, namely the Holland procedure and the Finner procedure [17,18]. It is noted that the best results listed in Tables 2, 6–9, 15 and 20 are shown in bold and the adjusted p-value shown in bold in Tables 4, 12–13, 17 and 22 mean that there exist statistically significant difference between CSPSO and the compared algorithm.

4.1. Benchmark functions

There are 17 widely used benchmark functions used in the first three experimental tests. These test functions listed in Table 1 are classified into three groups. The first group includes four unimodal functions, which are used to test the convergence speed of the CSPSO algorithm. The second group consists of five multimodal functions with many local optima, which are used to test the global search ability of the CSPSO algorithm. The unimodal and multimodal functions are available in [9,41,71,75]. The rest of the benchmark functions is more complicated shifted and rotated functions defined in [62], which can be used to validate the comprehensive performance of the CSPSO algorithm.

In Table 1, $f_{\min}$ gives the global optimal value (column 6) and x^* gives the global optimal solution (column 5). The column "Search Range" defines the lower and upper bounds of the definition domain in all dimensions. Moreover, biased initializations (column 4) are used for the functions whose global optima are at the center of the search range.

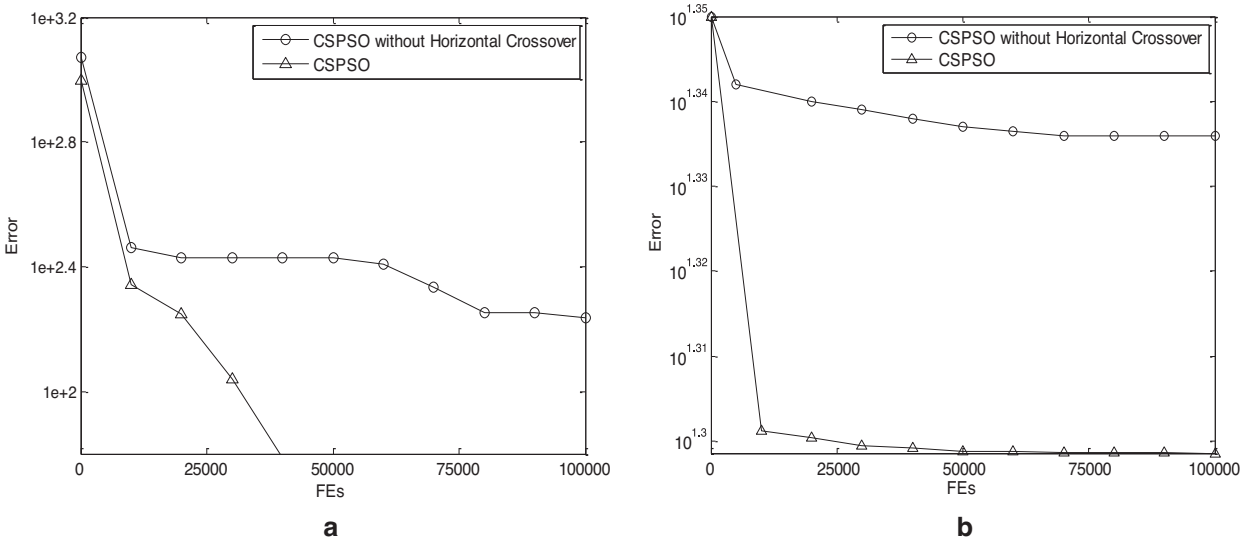


Fig. 1. Convergence curves of the CSPSO with and without the horizontal crossover on shifted and rotated functions (a) f_{13} (Shifted Rotated Rastrigin's) and (b) f_{16} (Shifted Rotated Ackley).

It is noted that all the experimental tests in this work were implemented on Matlab2010 platform and the simulations were executed on a PC with a Core (TM) 2 Duo CPU P8400 running at 2.26 GHz with memory capacity of 1.94GB under Windows XP Professional Operating System.

4.2. The effect of crisscross search on CSPSO

4.2.1. The effect of horizontal crossover on CSPSO

To validate the role of the horizontal crossover search in CSPSO, two benchmark functions f_{13} (Shifted Rotated Rastrigin's) and f_{16} (Shifted Rotated Ackley) shown in Table 1 were used for comparative analysis. In the experiments, two versions of CSPSOs were applied to the optimization on two 30-dimensional functions: f_{13} and f_{16}. They are the complete CSPSO described in Section 3.3 and the incomplete CSPSO in which the horizontal crossover has been removed, respectively. In this test, the number of particles was set to 20, the maximum number of function evaluations (*Max*FEs) in a run is set to 10^5 and the vertical crossover probability P_v. was set to 0.8. The typical convergence curves of the CSPSO with and without the horizontal crossover search on f_{13} and f_{16} are illustrated in Fig. 1(a) and (b), respectively. In the figure, the vertical ordinate is denoted by the error that is the absolute difference between the global optimum and the optimal result achieved by CSPSO with or without the horizontal crossover search. According to Fig. 1, it can be clearly seen that, on the complex shifted and rotated functions f_{13} and f_{16}, the CSPSO without the horizontal crossover search compares poorly with the complete CSPSO in terms of convergence accuracy and rate. It indicates that the horizontal crossover search plays an important role in addressing such complex problems as the shifted and rotated functions.

4.2.2. The effect of vertical crossover on CSPSO

As a unique constituent element of CSPSO, the vertical crossover search plays an important role in preventing CSPSO from trapping in local minima when addressing the multimodal problems. To investigate the effect of the vertical crossover search on CSPSO, we done the comparative experiments on two benchmark functions: the multimodal Schwefel's function and the multimodal Rastrigin function, which are listed in Table 1 as f_9 and f_5, respectively. The experimental parameters are the same as that used in the behavioral analysis of the horizontal crossover search. The typical convergence curves of the CSPSO with and without the vertical crossover search on f_9 and f_5 are illustrated in Fig. 2(a) and (b), respectively. According to Fig. 2, we observe that the CSPSO without the vertical crossover search (i.e., leaving only the modified PSO search and the horizontal crossover search) suffers the premature convergence problem in the optimizations on both of the multimodal functions of f_9 and f_5. The true reason lies in the fact that, on f_9, there are about 27% whole dimensions of the population that are struck stagnant, and on f_5, the number of the stagnant dimensions accounts for 40% in all 30 dimensions. This phenomenon can also explain why most of other population-based stochastic algorithms are possibly trapped in local optima. Compared with CSPSO without the vertical crossover search, we find that CSPSO (with the vertical crossover search) can easily escape from the local optima and converge to the global optimum 0 before 50,000 FEs on f_9 and f_5.

Another interesting question is how to determine the parameter P_v. In order to investigate the influence of P_v on CSPSO, empirical studies were carried out on the unimodal Rosenbrock's function and the multimodal Rotated Rastrigin's function listed in Table 1 as f_3 and f_{11}. Different values of P_v from 0 to 1 were tested. The results of the investigation are shown in Fig. 3. In the

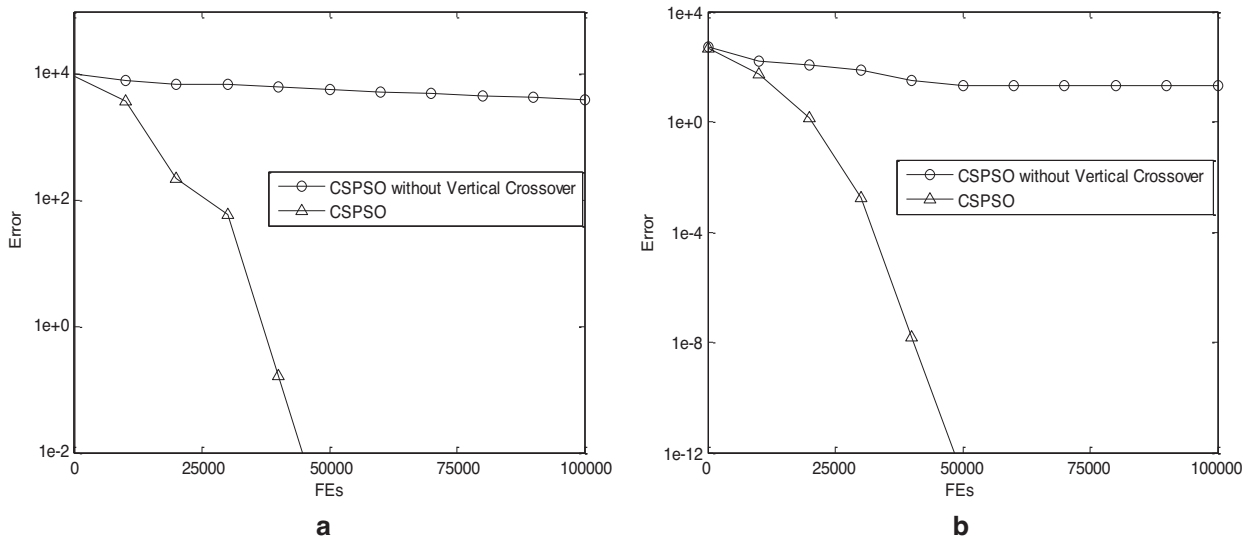


Fig. 2. Convergence curves of CSPSO with and without the horizontal crossover on multimodal functions (a) f_9 (Schwefel's) and (b) f_5 (Rastrigin).

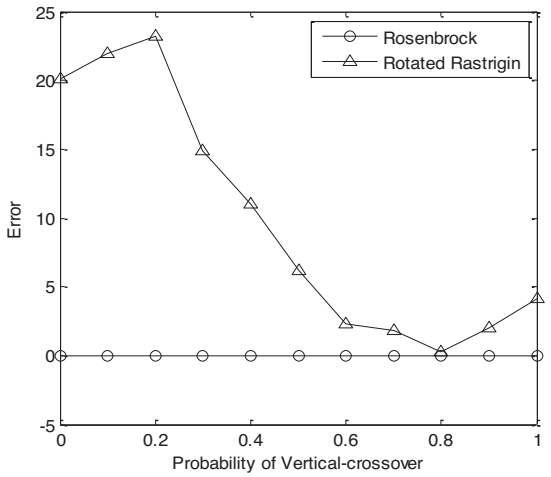


Fig. 3. Influence of the vertical crossover probability on the performance of CSPSO on functions (a) f_3 (Rosenbrock's) and (b) f_{11} (Rotated Rastrigin's).

figure, the vertical ordinate is the average error in 30 independent runs under different crossover probability levels. On the unimodal function f_3, the results reveal that the parameter P_v has little influence on the solution accuracy. However, on the multimodal function f_{11}, it significantly affects the optimal results. Here, a value of P_v around 0.8 offers the best performance. As one vertical crossover operation only reproduces a single offspring, therefore, the value of $P_v = 0.8$ indicates that 80% of the dimensions participate in the vertical crossover operation, but the actual number of dimensions to be mutated occupies only 40%. In view of our observations that about 20–40% of the whole dimensions in the swarm may be simultaneously trapped into stagnancy when optimizing the multimodal functions, we suggest to set P_v in the range [0.4,0.8] in order to make the stagnant dimensions jump out of the local optima. However, on the unimodal functions, the parameter P_v is recommended to be always set to zero so that the number of function evaluations (FEs) can fall by a third.

4.3. Comparisons of CSPSO with PSO and PSO + CSO

In order to investigate how much the improvement of the results depends on the crisscross search and on the choice of using only one population, we conducted a comparative study between CSPSO and the original PSO as well as between CSPSO and PSO + CSO. The main difference between CSPSO and PSO + CSO is that the former always maintains only one population of *pbests* while the latter consists of two independent algorithms (i.e., PSO and CSO). Therefore, PSO + CSO has two populations which operate alternatively at each iteration. In the optimization process of PSO + CSO, the population of PSO is updated by *pbests* and *gbest* according to Eqs. (1) and (2) rather than Eqs. (7) and (8). Subsequently, the updated population directly acts as

Table 2
Results comparisons between CSPSO, PSO and PSO + CSO.

Function	CSPSO	PSO + CSO	PSO
f_1	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	2.78E−161 ± 5.67E−160
f_2	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	2.03E−84 ± 7.17E−85
f_3	**0.00E+00** ± 0.00E+00	1.54E+01 ± 8.86E+00	2.41E+01 ± 1.69E+01
f_4	**0.00E+00** ± 0.00E+00	1.54E+01 ± 8.86E+00	1.85E−13 ± 1.91E−13
f_5	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	4.09E+01 ± 1.26E+01
f_6	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	2.65E+01 ± 8.61E+00
f_7	**0.00E+00** ± 0.00E+00	4.97E−15 ± 3.17E−15	8.20E−01 ± 2.05E+00
f_8	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	2.11E−02 ± 2.23E−02
f_9	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 2.97E−10	8.14E+03 ± 1.89E+03
f_{10}	**3.99E+00** ± 2.74E+00	1.35E+01 ± 7.95E+00	3.40E+01 ± 4.74E+01
f_{11}	**6.56E+00** ± 6.80E+00	1.60E+01 ± 1.38E+01	4.11E+01 ± 1.07E+01
f_{12}	**3.64E+04** ± 3.22E+04	8.03E+04 ± 9.49E+04	4.66E+05 ± 8.24E+04
f_{13}	**4.87E+01** ± 1.92E+01	5.21E+01 ± 8.93E+00	2.64E+02 ± 5.74E+01
f_{14}	**0.00E+00** ± 0.00E+00	2.50E−02 ± 1.20E−02	3.41E+02 ± 2.24E+02
f_{15}	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	4.65E+00 ± 4.95E+00
f_{16}	**2.06E+01** ± 2.20E−01	2.57E+01 ± 2.40E+00	2.09E+01 ± 7.00 E−01
f_{17}	**2.52E−26** ± 8.49E−21	5.66E−20 ± 8.11E−20	4.21E+01 ± 5.78E+01
$w/t/l$	–	10/7/0	17/0/0

Table 3
Average ranks achieved by the Friedman test for CSPSO, PSO + CSO and PSO. The highest rank is shown in bold.

Algorithms	Average ranks
CSPSO	**1.206**
PSO + CSO	1.912
PSO	2.882
Statistic	24.0737
p-value	0

Table 4
Adjusted p-values between CSPSO and two other PSO variants achieved by the Holland and Finer procedures.

CSPSO vs.	Holland	Finer
PSO	**0**	**0**
PSO + CSO	0.086	0.086

the father population of CSO without requiring the competition operation. Obviously, The CSO in the hybrid PSO + CSO is just responsible for producing *pbests* and *gbest* and feeding them to PSO at each iteration.

In this experimental test, the number of dimensions of all the test functions was set to $D = 30$, the population size was set to $M = 20$, and the maximum number of function evaluations in a run for all the test functions was set to $Max\text{FEs} = 2 \times 10^5$. To reduce statistical errors, each test was repeated 25 times independently. Moreover, biased initializations (column 5 in Table 1) were used for the functions whose global optimum is at the center of the search range. In this test, the inertia weight $\omega = 0.4$, the acceleration constants $c_1 = c_2 = 2.0$ and the vertical crossover probability $P_v = 0.8$.

The experimental results on the mean error values and the standard deviations are reported in Table 2. The comparison results between CSPSO and other two algorithms are summarized as "$w/t/l$" in the last row of the table, which indicates that CSPSO wins in w functions, ties in t functions and loses in l functions, compared with its peers [68,69]. The statistical results of the Friedman and post hoc tests are reported in Tables 3 and 4, respectively.

Compared with the original PSO, it can be seen that CSPSO has overwhelming advantage in terms of solution accuracy and standard deviations on all the test functions listed in Table 1. In this test, we observed that the original PSO was trapped in local minima on a majority of the test problems. However, CSPSO could reach the global optima without any accuracy error on all the unimodal and multimodal functions and even on more complex shifted and rotated functions like Shifted Griewank and Rotated Griewank. From this point of view, indeed, the crisscross search technique can significantly improve PSO's global search ability.

Similar to CSPSO, PSO + CSO (with two populations) also achieves good performance on $f_1, f_2, f_5, f_6, f_8, f_9$ and f_{15}. However, CSPSO performs better on the rest of the test functions. It wins in ten functions and ties in seven functions, compared with PSO + CSO.

To further compare the performance of multiple algorithms on the test suite, we conducted the Friedman test. Table 3 shows the average ranks of CSPSO, PSO + CSO and PSO on functions f_1–f_{17}. The highest rank is shown in bold. As seen, the performances

Table 5
Parameter setting of the compared PSO variants.

Algorithms	Year	Parameters settings
TLPSO-IDL [41]	2013	$\omega: 0.9-0.4, c_1: 2-1.5, c_2: 1-1.5, K_1 = 0.4, K_2 = 0.8, T_{size,max}: 0.5 \times S, R_{\max}: 0.1-1.0, m = 5$
FLPSO-QIW [63]	2011	$\omega: 0.9-0.2, c_1: 2-1.5, c_2: 1-1.5, m = 1, P_i: [0.1, 1], K_1 = 0.1, K_2 = 0.001, \sigma_1 = 1, \sigma_2 = 0$
RPPSO [80]	2011	$\omega: 0.9-0.4; c_{\text{large}} = 6; c_{\text{small}} = 3$
APSO [77]	2009	$\omega: 0.9-0.4, c_1 = c_2 = 2.0, c_1 + c_2: [3.0, 4.0], \delta: [0.05, 0.1], \sigma_{\max} = 1.0, \sigma_{\min} = 0.1$
FPSO [49]	2009	$\omega: 0.9-0.4, \sum c_i = 4.1$
ALC-PSO [12]	2013	$\omega: 0.9-0.4, T = 2, \Theta_0 = 60$
DMS-PSO [79]	2005	$\omega: 0.9-0.4, m = 3, R = 5$
CFPSO [13]	2008	$\omega: 0.9-0.4, c_1 = c_2 = 2.0, IP = 20$
Proposed CSPSO	–	$\omega = 0.4, c_1 = c_2 = 2.0, P_v = 0.8$

Table 6
Results of comparison between CSPSO and the first group of compared algorithms.

Functions	CSPSO	CSPSO*	TLPSO-IDL [41]	FLPSO-QIW [63]	RPPSO [80]	APSO [77]	FPSO [49]
f_1	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	4.07E−48 ± 1.12E−47	1.74E−02 ± 4.67E−02	1.70E−01 ± 1.28E−01	1.92E+02 ± 1.21E+02
f_3	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	2.46E+01 ± 5.07E−01	2.47E+01 ± 7.86E−01	3.23E+01 ± 1.69E+01	3.07E+01 ± 1.35E+01	3.99E+01 ± 6.48E+00
f_4	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	5.27E+01 ± 2.66E+01	6.46E+01 ± 3.83E+01	3.87E+02 ± 1.39E+02	2.51E+03 ± 1.44E+03
f_5	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	2.64E+00 ± 1.61E+00	2.19E+00 ± 6.31E+00	1.02E+00 ± 1.01E+00	2.67E+01 ± 1.19E+01
f_6	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	3.70E+00 ± 2.14E+00	2.10E+00 ± 5.15E+00	6.15E−02 ± 1.82E−01	2.02E+01 ± 1.04E+01
f_7	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	4.77E−14 ± 3.94E−14	8.29E−01 ± 1.20E+00	6.53E−02 ± 3.34E−02	3.58E+00 ± 1.66E+00
f_8	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	1.97E−03 ± 1.97E−03	1.76E−02 ± 1.76E−02	1.90E−01 ± 1.90E−01	2.46E+00 ± 2.46E+00
f_{11}	**0.00E+00** ± 0.00E+00	6.56E+00 ± 6.80E+00	1.02E+01 ± 3.12E+01	4.55E+01 ± 8.88E+00	2.80E+01 ± 2.43E+01	1.22E+02 ± 1.62E+01	1.08E+02 ± 2.23E+01
f_{14}	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	4.85E−03 ± 4.12E−03	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	6.60E+02 ± 3.00E+02
f_{15}	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	1.40E+00 ± 4.35E−01	2.43E+01 ± 5.69E+01	1.01E+02 ± 7.50E+01	1.12E+01 ± 1.01E+01
f_{16}	**1.96E+01** ± 4.92E−02	2.03E+01 ± 2.20E−01	2.08E+01 ± 1.38E−01	2.10E+01 ± 5.25E−02	2.10E+01 ± 4.06E−02	2.06E+01 ± 1.24E−01	2.10E+01 ± 5.19E−02
f_{17}	2.85E−26 ± 6.99E−20	**2.52E−26** ± 8.49E−21	6.13E+00 ± 9.84E+00	1.56E+02 ± 7.97E+01	5.30E+02 ± 7.26E+02	3.33E+02 ± 1.73E+02	1.09E+04 ± 6.81E+03
w/t/l	–	2/9/1	4/8/0	12/0/0	11/1/0	11/1/0	12/0/0

CSPSO*: biased initializations (column 4 in Table 1) are used for the functions whose global solution point is at the center of the search range.

of the three algorithms rank as follows: CSPSO, PSO + CSO and PSO. The highest average rank is obtained by the CSPSO algorithm. It demonstrates that CSPSO is the best one among the tree PSO variants. The statistic and *p*-value are also shown in Table 3. The critical value is 5.991 assuming that the level of significance $\alpha = 0.05$, so the null-hypothesis is rejected. The *p*-value is 0, so the Friedman test strongly suggests the existence of significant differences among the algorithms considered.

For pairwise comparison between CSPSO and the other two algorithms, we conducted a post hoc test by using the Holland and Finer procedures. Table 4 shows the adjusted *p*-values when comparing between CSPSO and the other two algorithms. The adjusted *p*-value shown in bold mean that there exist statistically significant difference between CSPSO and the compared algorithms. From the results, it can be seen that CSPSO is significantly better than PSO. Although CSPSO is not significantly better than PSO + CSO, CSPSO performs better than PSO + CSO according to the average ranks shown in Table 3.

4.4. Comparison of CSPSO with several state-of-the-art PSO variants

In this section, several state-of-the-art PSO variants listed in Table 5 are used for comparisons. The related parameters in the PSO variants are set according to the literature. TLPSO-IDL [41] is a two-layer PSO with intelligent division of labor module. FLPSO-QIW [63] uses feedback learning mechanism with quadratic inertia weight in the optimization process. RPPSO [80] is a novel particle swarm optimization with random position. APPSO [77] adopts an adaptive mechanism in the searching. FPSO [49] combines a number of algorithmic components in the original PSO. ALC-PSO [12] is a recently developed PSO variant using a novel aging mechanism. DMS-PSO [79] is a hybrid dynamic multi-swarm PSO algorithm with Nelder–Mead simplex search method. CFPSO [13] adopts the catfish effect to improve the solution quality. In the proposed CSPSO algorithm, the vertical crossover probability is set to $P_v = 0.8$.

In order to make a fair comparison, according to the compared PSO variants, the number of dimensions of all the test functions was set to $D = 30$, the population size was set to $M = 20$, and the maximum number of function evaluations in a run for all the test functions was set to $Max\text{FEs} = 1 \times 10^5$. To reduce statistical errors, each test was repeated 30 times independently.

It is noted that Table 1 lists most of the test functions used by the compared PSO variants. However, there exist small discrepancies of choosing test functions among them. For example, the compared algorithms TLPSO-IDL, FLPSO-QIW, RPPSO, APSO and FPSO adopted the test functions: f_1, f_3–f_8, f_{11} and f_{14}–f_{17}, but the functions f_2, f_9–f_{10} and f_{12}–f_{13} are not available in the references. Likewise, ALC-PSO, DMS-PSO and CFPSO used the test functions: f_1–f_3, f_5–f_{10}, f_{12}–f_{13} and f_{17}, but f_4, f_{11}, f_{14}–f_{16} are not available. Therefore, the compared algorithms are divided into two groups and all the experimental results are reported in Tables 6–13.

Tables 6 and 7 show the results of mean error values and standard deviations (SD), which are used to test CSPSO's convergence accuracy. The comparison results between CSPSO and other two algorithms are summarized as "*w*/*t*/*l*" in the last row of the table, which indicates that CSPSO wins in *w* functions, ties in *t* functions and loses in *l* functions, compared with its peers. To test the searching efficiency, the SP values, which are defined as the quotient of the average number of function evaluations needed by an algorithm to reach the pre-specified thresholds and the success rate (Sus(%)), are reported in Tables 8 and 9. In case that one

Table 7
Results of comparison between CSPSO and the second group of compared algorithms.

Functions	CSPSO	CSPSO*	ALC-PSO [12]	DMS-PSO [79]	CFPSO [13]
f_1	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	1.67E−161 ± 8.20E−161	1.95E−54 ± 8.43E−54	**0.00E+00** ± 0.00E+00
f_2	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	1.16E−90 ±04.14E−90	2.39E−27 ± 1.19E−26	**0.00E+00** ± 0.00E+00
f_3	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	7.61E+00 ± 6.66E+00	1.95E+01 ± 1.20E+01	2.24E+01 ± 5.40E−01
f_5	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	2.53E−14 ± 1.38E−14	2.78E+01 ± 7.57E+00	1.59E+00 ± 9.61E−01
f_6	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	1.25E−11 ± 6.75E−11	3.15E+01 ± 6.10E+00	2.99E+01 ± 3.20E−01
f_7	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	1.15E−14 ± 2.94E−14	9.23E−15 ± 1.79E−15	**0.00E+00** ± 0.00E+00
f_8	**0.00E+00** ± 0.00E+00	**0.00E+00** ± 0.00E+00	1.22E−02 ± 1.58E−02	1.07E−02 ± 1.60E−02	1.55E−02 ± 2.05E−02
f_9	**0.00E+00** ± 1.92E−02	**0.00E+00** ± 0.00E+00	2.10E+01 ± 5.41E+01	2.94E+03 ± 4.78E+02	6.34E+03 ± 2.30E+03
f_{10}	**7.90E−01** ± 1.68E+00	3.99E+00 ± 2.74E+00	2.18E+01 ± 4.04E+01	5.03E+01 ± 7.06E+01	1.31E_02 ± 1.80E+01
f_{12}	**3.81E+04** ± 2.24E+04	3.94E+04 ± 3.22E+04	8.88E+05 ± 4.15E+05	6.43E+06 ± 2.78E+06	1.13E+06 ± 1.06E+06
f_{13}	**4.88E+01** ± 1.92E+01	**4.88E+01** ± 1.92E+01	1.12E+02 ± 2.84E+01	5.40E+01 ± 8.42E+00	8.59E+01 ± 2.03E+01
f_{17}	2.85E−26 ± 6.99E−20	**2.52E−26** ± 8.49E−21	7.34E−10 ± 2.73E−12	2.06E+01 ± 8.10E−01	1.43E+00 ± 1.32E+00
$w/t/l$	–	2/9/1	12/0/0	12/0/0	9/3/0

Table 8
Convergence speed and success rate comparisons between CSPSO and the first group of compared algorithms.

Functions		ε	CSPSO	TLPSO-IDL [41]	FLPSO-QIW [63]	RPPSO [79]	APSO [77]	FPSO [49]
f_1	Suc(%)	1e−6	**100**	100	100	76.67	0.00	0.00
	SP		**4.78E+03**	5.82E+03	2.41E+04	9.55E+03	Inf	Inf
f_3	Suc(%)	1e−2	**100**	0.00	0.00	0.00	0.00	0.00
	SP		**1.07E+03**	Inf	Inf	Inf	Inf	Inf
f_4	Suc(%)	1e−6	**100**	100	0.00	0.00	0.00	0.00
	SP		**5.07E+03**	1.12E+04	Inf	Inf	Inf	Inf
f_5	Suc(%)	1e−2	100	**100**	13.33	73.33	0.00	0.00
	SP		5.76E+04	**1.15E+04**	5.78E+05	3.33E+04	Inf	Inf
f_6	Suc(%)	1e−2	**100**	100	0.00	66.67	30.0	0.00
	SP		**8.60E+03**	1.57E+04	Inf	3.24E+04	8.37E+05	Inf
f_7	Suc(%)	1e−2	**100**	100	100	40	0.00	10.0
	SP		**5.57E+03**	6.53E+03	1.88E+04	1.11E+05	Inf	5.35E+05
f_8	Suc(%)	1e−2	**100**	100	93.33	83.33	0.00	3.33
	SP		**3.13E+03**	4.90E+03	2.33E+04	8.14E+03	Inf	4.06E+06
f_{11}	Suc(%)	−329.99	**100**	90	0.00	33.33	0.00	0.00
	SP		**3.92E+04**	4.76E+04	Inf	4.23E+04	Inf	Inf
f_{14}	Suc(%)	1e−2	100	100	100	**100**	100	6.67
	SP		1.33E+03	6.64E+02	2.11E+04	**3.12E+02**	8.14E+02	4.94E+03
f_{15}	Suc(%)	1e−2	**100**	100	0.00	73.33	0.00	0.00
	SP		**1.45E+03**	5.84E+03	Inf	1.02E+04	Inf	Inf
f_{16}	Suc(%)	1e−2	0.00	0.00	0.00	0.00	0.00	0.00
	SP		Inf	Inf	Inf	Inf	Inf	Inf
f_{17}	Suc(%)	−440	**100**	–	0.00	0.00	0.00	100
	SP		**1.13E+04**	–	Inf	Inf	Inf	4.70E+04

Suc(%): the percentage of trial runs successfully reaching acceptable accuracy.
SP: denotes the quotient of the average number of function evaluations needed by an algorithm to reach the pre-specified thresholds and the Suc%.
ε: stands for the acceptable threshold value achieved by the optimization algorithms in a trial.

algorithm fails to solve the problem (i.e., Suc(%) = 0) the SP is set to "Inf". The ε value in bracket in Tables 8 and 9 stands for the acceptable threshold value achieved by the optimization algorithms in a trial. The trial is considered successful if and only if the mean error value achieved by an algorithm is equal to or less than the acceptable value. Because the compared algorithms might adopt different ε values, for a fair comparison, the same threshold values are adopted according to the compared algorithms. To determine the statistical difference and pairwise comparisons between CSPSO and its peers, we also performed the Friedman test and a post hoc test. The statistical results are listed in Tables 10–13. To exhibit the convergence characteristic of the compared PSO algorithms, a dozen convergence curves are illustrated in Fig. 4. In this figure, the ordinate is denoted by the error, which is the difference between the best optimal value found by an algorithm in a trial and the global optimal value $f_{\min}$. It is noted that the data of the compared algorithms are exacted directly from the literature.

According to the experimental results, it can be clearly observed that CSPSO has obvious superiority over most of the other PSO variants in terms of solution accuracy and convergence speed. The further performance comparisons in detail are described as follows.

4.4.1. Unimodal functions

From the results in Tables 6 and 7, it is observed that only the proposed CSPSO can always converge to the global minimum point (zero) without any accuracy error on all the unimodal functions. The other PSO variants involved in the experiments also

Table 9
Convergence speed and success rate comparisons between CSPSO and the second group of compared algorithms.

Functions		ε	CSPSO	ALC-PSO [12]	DMS-PSO [79]	CFPSO [13]
f_1	Suc(%)	1e−2	**100**	100	100	100
	SP		**5.39E+03**	7.49E+03	9.15E+04	4.76E+04
f_2	Suc(%)	1e−2	**100**	100	100	100
	SP		**6.45E+03**	8.18E+03	9.06E+04	2.44E+04
f_3	Suc(%)	1e−2	**100**	100	100	100
	SP		**2.53E+03**	5.35E+03	8.60E+04	3.01E+04
f_5	Suc(%)	1e+1	**100**	100	0.00	100
	SP		**5.23E+04**	7.42E+04	Inf	9.25E+04
f_6	Suc(%)	1e+1	**100**	100	0.00	0.00
	SP		**8.35E+03**	5.89E+04	Inf	Inf
f_7	Suc(%)	1e−2	**100**	100	100	80
	SP		**7.58E+03**	5.89E+04	9.98E+04	3.45E+04
f_8	Suc(%)	1e−2	**100**	60	70	33.33
	SP		**6.13E+03**	1.70E+04	1.39E+05	1.95E+05
f_9	Suc(%)	−10569.48	**100**	100	33.33	0.00
	SP		**3.39E+04**	4.70E+04	2.64E+05	Inf
f_{10}	Suc(%)	490	**100**	90	80	60
	SP		**3.22E+04**	3.78E+04	1.53E+05	2.11E+05
f_{12}	Suc(%)	1e−7	**100**	100	86.7	0.00
	SP		**7.42E+03**	9.28E+03	1.12E+05	Inf
f_{13}	Suc(%)	−130	**100**	100	100	100
	SP		**2.32E+04**	1.71E+04	2.95E+04	6.45E+04
f_{17}	Suc(%)	−350	**100**	100	100	100
	SP		**7.87E+03**	3.07E+04	1.22E+05	1.86E+04

Table 10
Average ranks achieved by the Friedman test for CSPSO, TLPSO-IDL, FLPSO-QIW, APSO, RPPSO and FPSO. The highest rank is shown in bold.

Algorithms	Ranks
CSPSO	**1.46**
TLPSO-IDL [41]	1.88
FLPSO-QIW [63]	3.67
APSO [77]	4.13
RPPSO [80]	4.21
FPSO [49]	5.67
Statistic	45.177
p-value	0

Table 11
Average ranks of CSPSO, ALC-PSO, CFPSO and DMS-PSO achieved by the Friedman tests. The highest rank is shown in bold.

Algorithms	Ranks
CSPSO	**1.33**
ALC-PSO [12]	2.54
CFPSO [13]	2.83
DMS-PSO [79]	3.29
Statistic	17.286
p-value	0.001

Table 12
Adjusted *p*-values achieved by the Holland and Finer procedures between CSPSO and the first group of compared algorithms.

CSPSO vs.	Holland	Finer
FPSO [49]	**0**	**0**
RPPSO [80]	**0**	**0**
APSO [77]	**0**	**0**
FLPSO-QIW [63]	**0.0784**	**0.04974**
TLPSO-IDL [41]	0.585	0.585

Table 13
Adjusted p-values achieved by the Holland and Finer procedures between CSPSO and the second group of compared algorithms.

CSPSO vs.	Holland	Finer
DMS-PSO [79]	**0.005988**	**0.005988**
CFPSO [13]	**0.008**	**0.005994**
ALC-PSO [12]	**0.027**	**0.027**

have a good performance on most of the unimodal functions. For example, CFPSO converges to zero on f_1 and f_2, TLPSO-IDL can also converge to zero on f_1 and f_4. But for the Rosenbrock's functions f_3, except for the proposed CSPSO, the other PSO variants are all trapped in local minima and the mean error values vary between 7.613 (ALC-PSO) and 39.9 (FPSO). This problem is actually not surprising in that the unimodal function f_3 has assumed somewhat characteristic of a multimodal function in high-dimensional case [54]. Nevertheless, it is easy for CSPSO to overcome the premature convergence and reach the global optimal value 0 in all 30 runs when addressing such problem as f_3.

As CSPSO needs to take three times of function evaluations (FEs) at every iteration because of the addition of the horizontal crossover search and the vertical crossover search, compared with PSO. Consequently, the convergence speed becomes the key issue of our concern. In Tables 8 and 9, the average number of function evaluations that the algorithms need to reach the acceptable optimal value ε in 30 runs are reported. We observe that CSPSO not only has powerful global search ability to find the global minima, but also has fast-converging advantage over the other PSO variants when optimizing unimodal problems. The results prove that CSPSO does not sacrifice the convergence speed because it enhances PSO's convergence accuracy by incorporating the crisscross search. Conversely, it uses less SP to reach the acceptable results on all the unimodal functions. The reliability statistical results are also reported in Tables 8 and 9. "Suc(%)" stands for the percentage of the successful runs that acceptable solutions can be found in all 30 runs. According to Tables 8 and 9, we observe that FLPSO-QIW, RPPSO, APSO and FPSO cannot reach the acceptable values on f_3 and f_4. APSO and FPSO fail to converge to preset accuracy on f_1. Conversely, CSPSO requires the least computation cost to solve all these unimodal problems within a preset accuracy.

The global search ability and fast-converging behavior of CSPSO can be further illustrated in Fig. 4(a)–(c), in which, the error is defined as the difference between the actual optimal result yielded by any algorithm and the global optimum. For Sphere function (a) and Schwefel 1.2 function (c), both CSPSO and TLPSO-IDL can reach the global optima within one hundred thousand function evaluations. According to the error convergence nature on Roesnbrock's function, CSPSO can easily converge to zero, however, all the other compared PSOs are trapped in local minima.

4.4.2. Multimodal functions

For multimodal functions f_5–f_9 with many local minima, the global minima become more difficult to locate. As CSPSO incorporates with crisscross search technique, it is natural to form three mutually reinforcing search operations including the modified PSO search, the horizontal crossover search, and the vertical crossover search. We expect that CSPSO can escape from the local minima and enhance PSO's performance on the multimodal functions. The experimental results on f_5–f_9 in Tables 6 and 7 prove that CSPSO does have powerful global search ability when addressing multimodal problems. Indeed, for the multimodal functions f_4–f_8, the results in Tables 6 and 7 show that both the proposed CSPSO and TLPSO-IDL can always converge to the global minima without any accuracy error in all 30 runs. Among the other PSO versions, only CFPSO and TLPSO-IDL can reach the global minimum on f_7.

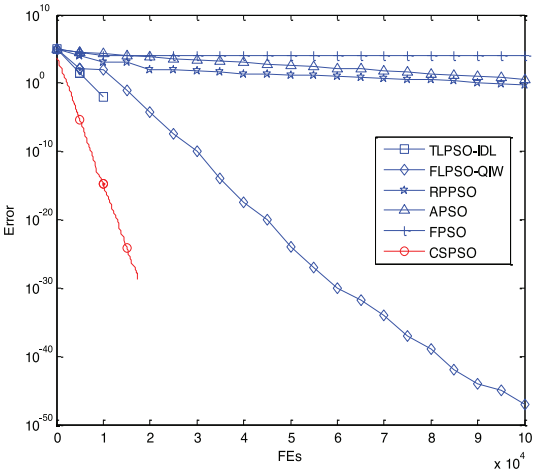
From Tables 8 and 9, we observe that only CSPSO can reach the preset accuracy in all 30 trials on all multimodal functions. The convergence speed of CSPSO in the multimodal problems is slightly inferior for producing only one second best SP values on function f_5, compared to the TLPSO-IDL algorithm. But for all other multimodal functions, the CSPSO exhibits the fastest convergence speed among all the compared algorithms.

Fig. 4(d)–(g) give the typical error convergence graphs of functions f_5–f_8 obtained by CSPSO and other PSO variants. It can be clearly seen that CSPSO can quickly converge to the global optima without accuracy error. The convergence speed results in Tables 8 and 9 further prove that CSPSO has a dominant advantage in terms of global search ability and convergence rate compared with other PSO versions.

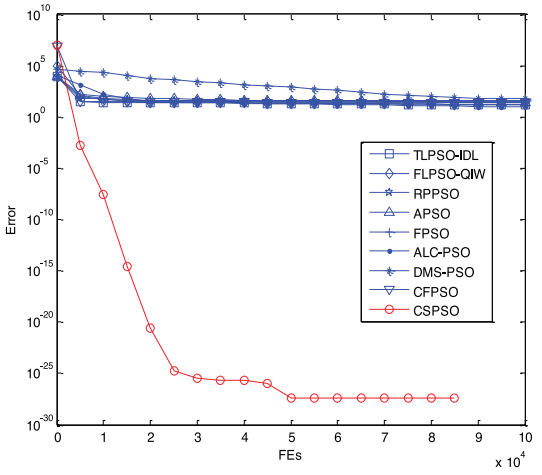
4.4.3. Shifted and rotated functions

In this experimental test, we used eight more complex test functions with coordinate shift and rotation to further test the convergence accuracy and speed of CSPSO. Among these shifted and rotated functions, here, f_{10}, f_{14} and f_{17} are the shifted functions of Rosenbrock's, Griewank, and Schwefel 1.2, respectively. f_{11} and f_{15} are the rotated functions of Rastrigin's and Griewank. f_{12}, f_{13}, and f_{16} are the shifted and rotated functions of High Conditioned Elliptic, Rastrigin's, Ackley, respectively. The coordinate rotation of unimodal or multimodal function will make the dimensions of these functions become non-separable, and thus the resulting problems become more difficult for a search algorithm to solve [11].

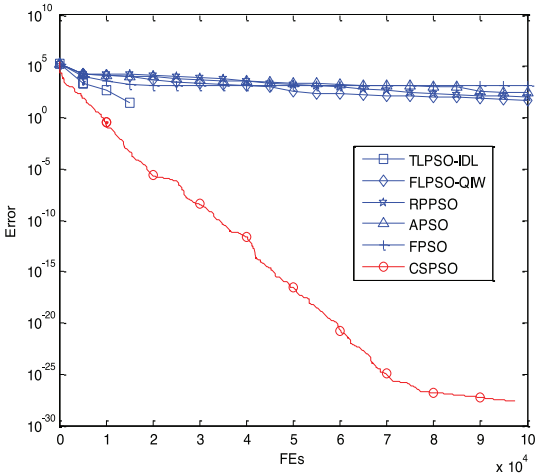
The experimental results on the shifted and rotated functions f_{10}–f_{17} are reported in Tables 6 and 7. From the experimental results, we observe that the mean optimal error values and the standard deviations on all the shifted and rotated functions obtained by CSPSO are far better than those yielded by the other PSO variants. It can be seen that CSPSO can converge to the



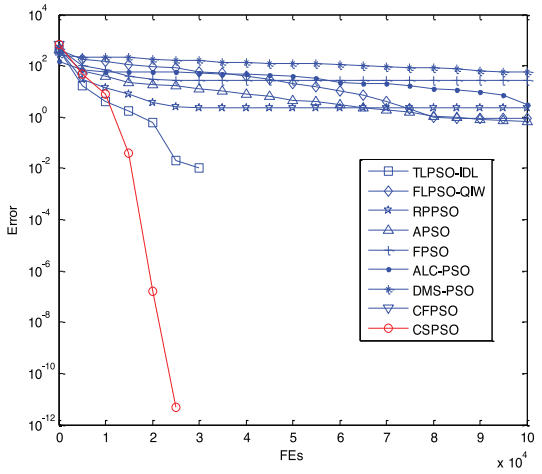
(a) (Sphere) (f_1)



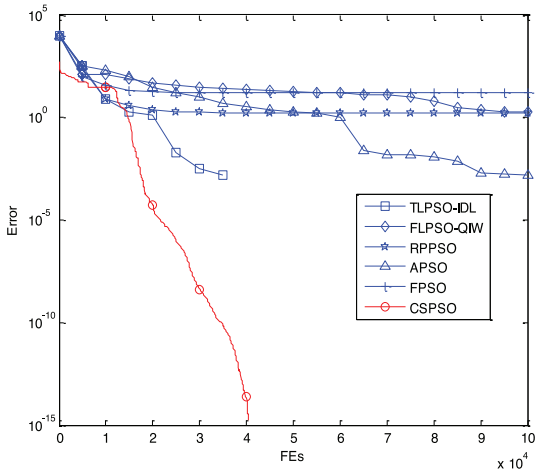
(b) (Rosenbrock's) (f_3)



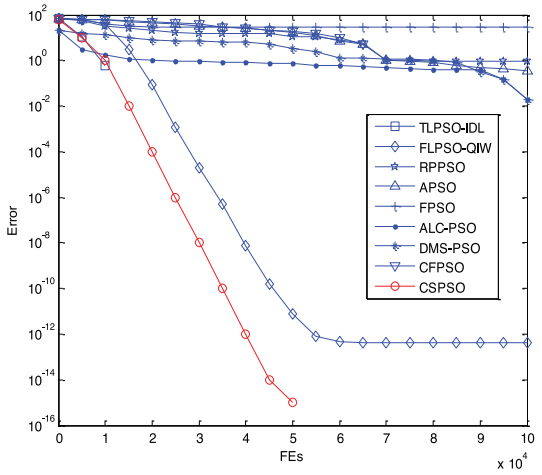
(c) (Schwefel 1.2) (f_4)



(d) (Rastrigin) (f_5)

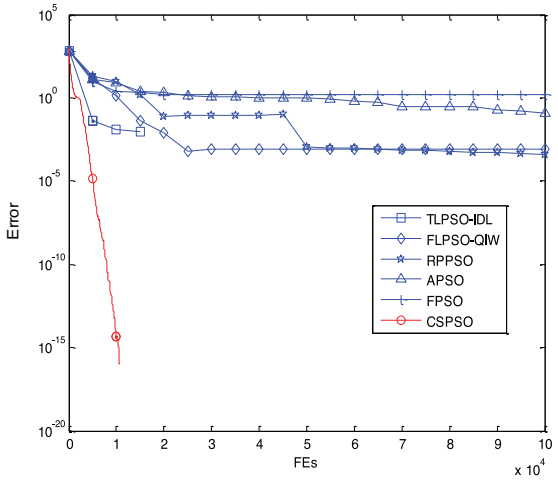


(e) (Noncontinuous Rastrigin) (f_6)

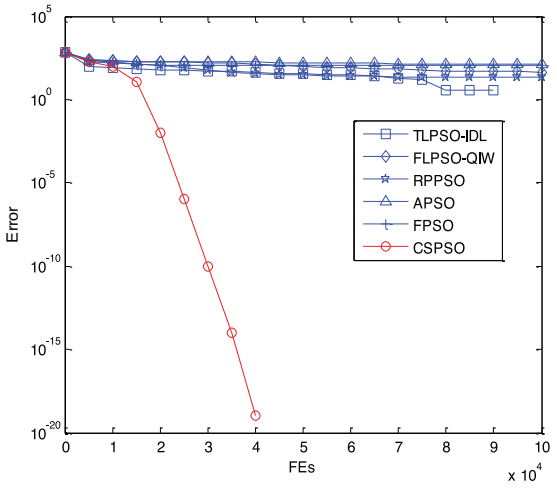


(f) (Ackley) (f_7)

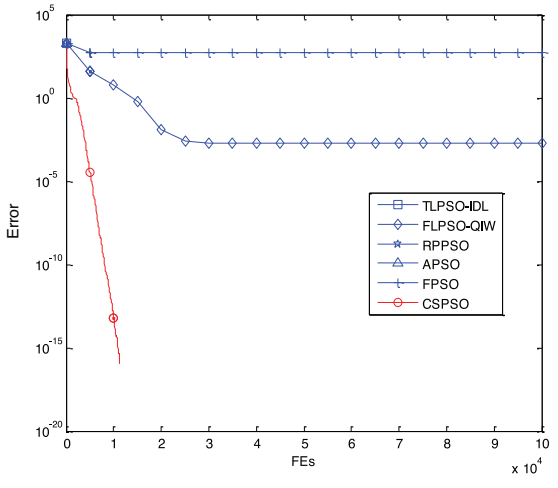
Fig. 4. The convergence graphs on functions (a) f_1 (Sphere), (b) f_3 (Rosenbrock), (c) f_4 (Schwefel 1.2), (d) f_5 (Rastrigin), (e) f_6 (Noncontinuous Rastrigin), (f) f_7 (Ackley), (g) f_8 (Griewank), (h) f_{11} (Rotated Rastrigin), (i) f_{14} (Shifted Griewank), (j) f_{15} (Rotated Griewank), (k) f_{16} (Shifted Rotated Ackley), (l) f_{17} (Shifted Schwefel's P1.2).



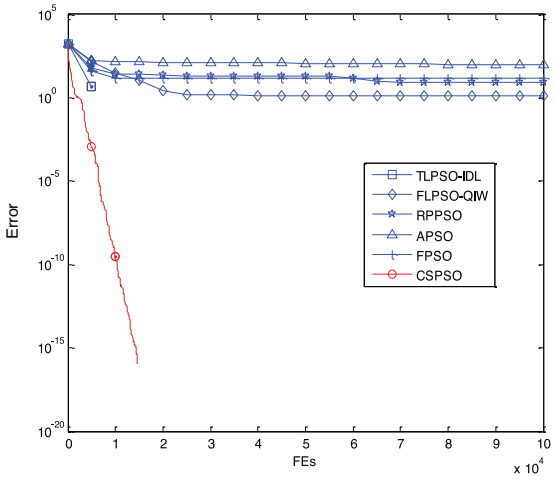
(g) (Griewank) (f_8)



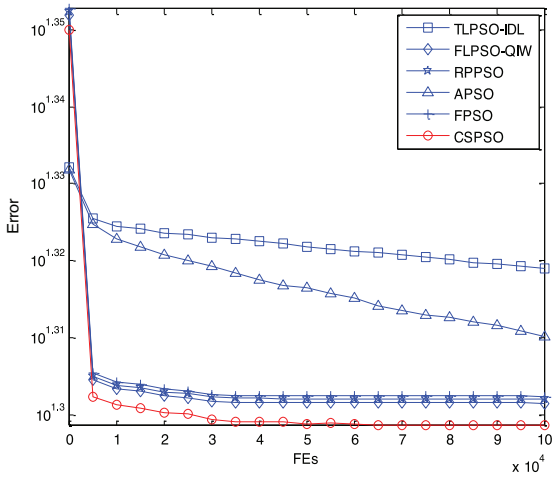
(h) (Rotated Rastrigin) (f_{11})



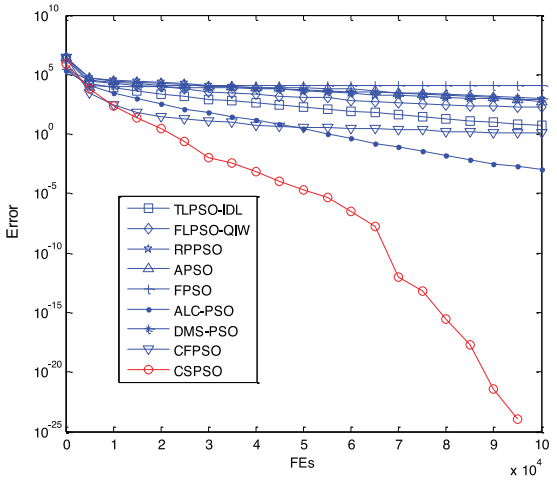
(i) (Shifted Griewank) (f_{14})



(j) (Rotated Griewank) (f_{15})



(k) (Shifted rotated Ackley) (f_{16})



(l) (Shifted Schwefel's P1.2) (f_{17})

Fig. 4. Continued

Table 14
Parameter setting of CSPSO, CLPSO, OLPSO-L and OLPSO-G.

Algorithms	Year	Parameters settings
OLPSO-L [78]	2011	$\omega: 0.9-0.4, c=2.0, G=5, V_{MAXd}=0.5\times$ Range
OLPSO-G [78]	2011	$\omega: 0.9-0.4, c=2.0, G=5, V_{MAXd}=0.5\times$ Range
CLPSO [39]	2006	$\omega: 0.9-0.4, c=1.49445, m=7, V_{MAXd}=0.2\times$ Range
CSPSO	–	$\omega: 0.4, c_1=c_2=2.0, P_i=0.8$

global minima without any accuracy error on f_{11}, f_{14} and f_{15}. TLPSO-IDL can also converge to zero on f_{14} and f_{15}. RPPSO and APSO can reach the global optimum on f_{14}, too. For the rest of functions, CSPSO exhibits better performance obviously.

In terms of SP and successful rate in the reliability test, among all the algorithms, CSPSO is the only one that can reach the acceptable result at 100% successful rates on all the shifted and rotated functions except for f_{16}. For the function f_{16}, although CSPSO failed to reach the acceptable solution, neither are the other PSO variants.

Fig. 4(h) and (l) give the typical error convergence graphs of functions $f_{11}, f_{14}, f_{15}, f_{16}$ and f_{17} obtained by CSPSO and other PSO variants. It can be seen that CSPSO outperforms the other PSO variants significantly. Especially, the proposed CSPSO can converges to zero at a very faster rate within one hundred thousand function evaluations on Rotated Rastrigin function, Shifted Griewank function and Rotated Griewank function.

In order to demonstrate that the proposed CSPSO can still work efficiently in an unsymmetrical searching range, we also used the biased initialization method (column 4 in Table 1) to initialize the population for those functions whose global solution point is at the center of the search range. The results obtained by CSPSO with biased initialization (represented as CSPSO* in Tables 6 and 7) show that for functions f_1–f_9, CSPSO* can also converge to the global optima in all 30 trials. For the rest of test functions, CSPSO* is slightly inferior to CSPSO, nevertheless, it still outperforms the other compared PSO variants.

4.4.4. Statistical analysis

According to the "$w/t/l$" results shown in Tables 6 and 7, CSPSO achieves better results than FLPSO-QIW, FPSO, DMS-PSO and ALC-PSO on all the functions. Compared with RPPSO and APSO, CSPSO wins in 11 functions, loses in only one function. For TLPSO-IDL, CSPSO wins in four functions, ties in eight functions. Compared with CFPSO, CSPSO wins in nine functions, loses in three functions.

To further compare the performance of multiple algorithms on the test suite, we conducted the Friedman test. Tables 10 and 11 show the average ranks of all the compared algorithms. The highest rank listed in Tables 10 and 11 is shown in bold. As seen in the results, the highest average rank is obtained by the CSPSO algorithm. It demonstrates that CSPSO is the best one among the eight PSO algorithms on the benchmark problems tested. The Friedman statistic and p-value reported in Tables 10 and 11 also reveal the existence of significant differences of results obtained by the compared algorithms.

Given that the level of significance $\alpha = 0.05$, the critical values for the two groups of compared algorithms are 11.071 and 7.815, respectively. According to the Friedman statistic values and p-values shown in Tables 10 and 11, it can see that there exist significant differences among the algorithms compared.

For pairwise comparisons between CSPSO and the other state-of-the-art PSO variants, we proceeded to conduct a post hoc test through the Holland procedure and the Finner procedure. The adjusted p-values obtained by comparing between CSPSO and the other PSO variants are shown in Tables 12 and 13. The adjusted p-values shown in bold mean that the differences between CSPSO and the PSO variants are significant. From the results, it can be seen that CSPSO is significantly better than the other PSO variants except for TLPSO-IDL in Table 12. Although CSPSO is not significantly better than TLPSO-IDL, CSPSO performs better than TLPSO-IDL according to the average ranks shown in Table 10.

This experimental finding strongly suggests that the better results achieved by CSPSO are statistically significant and are not obtained by chance.

4.5. Comparison of CSPSO with two popular PSO variants

In this section, we added orthogonal learning particle swarm optimization (OLPSO) [78] and comprehensive learning particle swarm optimization (CLPSO) [39] for comparison due to their popularity. OLPSO adopts an orthogonal learning strategy (OL) to guide particles to fly in better directions by constructing a much promising and efficient exemplar. In OLPSO, the OL strategy is applied to both global and local versions of PSO, yielding the OLPSO-G and OLPSO-L algorithms, respectively. CLPSO uses a novel learning strategy whereby all other particles' historical best information is used to update a particle's velocity.

In this test, the related parameters used in the algorithms are listed in Table 14 and the benchmark functions used in Table 15 were selected as the same as CLPSO and OLPSO did in [39,78]. For all the algorithms listed in Table 14, the number of dimensions of all the test functions was set to $D = 30$, the population size was set to $M = 20$, and the maximum number of function evaluations in a run for all the test functions was set to $MaxFEs = 2 \times 10^5$. To reduce statistical errors, each test was repeated 25 times independently. Moreover, biased initializations (column 2 in Table 15) were used for the functions whose global solution point is at the center of the search range.

Table 15 presents the results of mean error values and standard deviations obtained in this case and the comparison of results between CSPSO and its peers are summarized as "$w/t/l$" in the last row of this table, which means that CSPSO wins in w functions,

Table 15
Results of comparison between CSPSO and two popular PSO algorithms.

Function	Initialization Range	CSPSO	CLPSO [39]	OLPSO-G [78]	OLPSO-L [78]
F_1 (Sphere)	$[-100,50]^D$	**0.00E+00** ± 0.00E+00	1.58E−12 ± 7.70E−13	4.12E−54 ± 6.34E−54	1.11E−38 ± 1.28E−38
F_2 (Schwefel's P2.22)	$[-10,5]^D$	**0.00E+00** ± 0.00E+00	2.51E−08 ± 5.84E−09	9.85E−30 ± 1.01E−29	7.67E−22 ± 5.63E−22
F_3 (Rosenbrock's)	$[-10,10]^D$	**0.00E+00** ± 0.00E+00	1.14E+01 ± 9.85E+00	2.15E+01 ± 2.99E+01	1.26E+00 ± 1.40E+00
F_4 (Noise)	$[-1.28,0.64]^D$	8.43E+00 ± 3.60E−01	**5.85E−03** ± 1.11E−03	1.16E−02 ± 4.10E−03	1.64E−02 ± 3.25E−03
F_5 (Schwefel's)	$[-500,500]^D$	**0.00E+00** ± 0.00E+00	3.82E−04 ± 1.28E−07	3.84E+02 ± 2.17E+02	3.82E−04 ± 0.00E+00
F_6 (Rastrigin)	$[-5.12,2]^D$	**0.00E+00** ± 0.00E+00	9.09E−05 ± 1.25E−04	1.07E+00 ± 9.90E−01	**0.00E+00** ± 0.00E+00
F_7 (Ackley)	$[-32,16]^D$	**0.00E+00** ± 0.00E+00	3.66E−07 ± 7.57E−08	7.98E−15 ± 2.03E−15	4.14E−15 ± 0.00E+00
F_8 (Griewank)	$[-600,200]^D$	**0.00E+00** ± 0.00E+00	9.02E−09 ± 8.57E−09	4.83E−03 ± 8.63E−03	**0.00E+00** ± 0.00E+00
F_9 (Generalized Penalized−1)	$[-50,25]^D$	**1.57E−32** ± 0.00E+00	6.45E−14 ± 3.70E−14	1.59 E−32 ± 1.03E−33	**1.57E−32** ± 2.79E−48
F_{10} (Generalized Penalized-2)	$[-50,25]^D$	1.34E−31 ± 0.00E+00	1.25E−12 ± 9.45E−13	4.39E−04 ± 2.20E−03	**1.35E−32** ± 5.59E−48
F_{11} (Rotated Schwefel)	$[-500,500]^D$	**2.46E+03** ± 3.87E+03	4.39E+03 ± 3.51E+02	4.00E+03 ± 6.08E+02	3.13E+03 ± 1.24E+03
F_{12} (Rotated Rastrigin's)	$[-5.12,2]^D$	**6.56E+00** ± 6.80E+00	8.71E+01 ± 1.08E+01	4.61E+01 ± 1.29E+01	5.34E+01 ± 1.34E+01
F_{13} (Rotated Ackley)	$[-32,16]^D$	9.95E−01 ± 2.2E−02	5.91E−05 ± 6.46E−05	7.69E−15 ± 1.78E−15	**4.28E−15** ± 7.11E−16
F_{14} (Rotated Griewank)	$[-600,200]^D$	**0.00E+00** ± 0.00E+00	7.96E−05 ± 7.66E−05	1.68E−03 ± 4.13E−03	4.19E−08 ± 2.06E−07
F_{15} (Shifted Rosenbrock's)	$[-100,100]^D$	**3.99E+00** ± 2.74E+00	1.30E+01 ± 1.35E+01	3.5E+03 ± 3.48E+01	2.60E+03 ± 2.40E+01
F_{16} (Shifted Rastrigin)	$[-5,5]^D$	4.55E+01± 9.75E+00	**0.00E+00** ± 3.39E−05	1.43E+00 ± 1.04E+00	**0.00E+00** ± 1.64E−14
$w/t/l$		–	13/0/3	13/0/3	9/3/4

Table 16
Average ranks of CSPSO, CLPSO, OLPSO-G and OLPSO-L achieved by the Friedman tests. The highest rank is shown in bold.

Algorithms	Ranks
CSPSO	**1.75**
OLPSO-L [78]	2.09
CLPSO [39]	3.06
OLPSO-G [78]	3.09
Statistic	13.929
p-value	0.003

Table 17
Adjusted p-values achieved by the Holland and Finer procedures between CSPSO and two popular PSO algorithms.

CSPSO vs.	Holland	Finer
CLPSO [39]	**0.03263**	**0.03264**
OLPSO-G [78]	**0.0396**	**0.03264**
OLPSO-L [78]	0.732	0.732

ties in t functions and loses in l functions, compared with its peers. And the statistical results obtained from the Friedman tests and the post hoc tests are listed in Tables 16 and 17.

The "$w/t/l$" values show that CSPSO achieves better results than CLPSO and OLPSO-G on 13 functions, while CLPSO and OLPSO-G performs better than CSPSO on three functions. Compared with OLPSO-L, CSPSO wins in nine functions, ties in three functions and loses in four functions.

As seen from Table 15, CSPSO surpasses the other three PSO versions on all the unimodal and multimodal functions F_1–F_{10} except for F_4 and F_{10}. For the rotated and shifted functions F_{11}–F_{16}, CSPSO can converge to zero in all 25 trials on F_{14}. CLPSO and OLPSO-L can also reach the global minimum on function F_{16}. However, CSPSO yields the best results among the four PSOs on functions F_{11}, F_{12}, F_{14} and F_{15}.

Table 16 shows the average ranks of CSPSO, CLPSO, OLPSO-G and OLPSO-L. The highest rank is shown in bold. As seen, the performance of the four algorithms can be sorted by average ranks into the following order: CSPSO, OLPSO-L, CLPSO and OLPSO-G. The highest average rank is obtained by the CSPSO algorithm. It demonstrates that CSPSO is the best one among the four PSO variants for comparison functions.

The statistic value and p-value shown in Table 16 are 13.929 and 0.003, respectively. The critical value for $\alpha = 0.05$ is 7.815. So the Friedman test strongly suggests the existence of significant differences among the four algorithms.

Table 17 shows the adjusted p-values obtained by comparing between CSPSO and the other algorithms through the post hoc tests. From the results listed in Table 14, we observe that since CSPSO is just slightly superior to the local version of CLPSO (i.e., CLPSO-L) on most of the test functions. Therefore, it is not surprising that the results achieved by the post hoc test cannot statistically detect the difference between CSPSO and CLPSO-L. Compared with CLPSO and the global version of CLPSO (i.e., CLPSO-G), the better solutions obtained by CSPSO are statistically significant.

Table 18
A set of 10 CEC 2005 benchmark functions.

Function	Name	D	$f_{\min}$
f_{cec05_1}	Shifted Sphere function	30	−450
f_{cec05_2}	Shifted Schwefel's Problem 1.2	30	−450
f_{cec05_3}	Shifted Rotated high conditioned elliptic function	30	−450
f_{cec05_4}	Shifted Schwefel's Problem 1.2 with Noise	30	−450
f_{cec05_5}	Schwefel's Problem 2.6 with global optimum on bounds	30	−310
f_{cec05_6}	Shifted Rosenbrock's function	30	390
f_{cec05_7}	Shifted Rotated Griewank's function	30	−180
f_{cec05_8}	Shifted Rotated Ackley's function	30	−140
f_{cec05_9}	Shifted Rastrigin's function	30	−330
$f_{cec05_{10}}$	Shifted Rotated Rastrigin's function	30	−330

Table 19
Parameter settings of the compared PSO variants.

Algorithms	Year	Parameters settings
DNSPSO [68]	2013	$\omega = 0.7298$, $c_1 = c_2 = 1.49618$, $k = 2$
DNSCLPSO [39]	2013	$\omega = 0.7298$, $c_1 = c_2 = 1.49618$, $k = 2$, $m = 7$, $V_{MAXd} = 0.2 \times$ Range
CLPSO [68]	2011	$\omega : 0.9 - 0.4$, $c_1 = c_2 = 1.49445$, $m = 7$, $V_{MAXd} = 0.2 \times$ Range
GOPSO [69]	2011	$\omega = 0.72984$, $c_1 = c_2 = 1.49618$, $V_{MAXd} = 0.5 \times$ Range
APSO [77]	2009	$\omega : 0.9 - 0.4$, $c_1 = c_2 = 2.0$, $c_1 + c_2 : [3.0, 4.0]$, $\delta : [0.05, 0.1]$, $\sigma_{\max} = 1.0$, $\sigma_{\min} = 0.1$
CPSO-H [66]	2006	$\omega : 0.9 - 0.4$, $c_1 = 1.49$, $c_2 = 1.49$, $V_{MAXd} = 0.5 \times$ Range
CSPSO	–	$\omega = 0.4$, $c_1 = c_2 = 2.0$, $P_v = 0.8$

Table 20
Results of comparison between CSPSO and other PSOs on 10 CEC 2005 benchmark functions.

Function	CSPSO	CPSO-H [66]	APSO [77]	GOPSO [69]	CLPSO [39]	DNSCLPSO [68]	DNSPSO [68]
f_{cec05_1}	**0.00E+00** ± 0.00E+00	4.26E−02 ± 1.25E+00	7.20E−14 ± 1.38E−13	5.86E−14 ± 6.37E−14	8.01E−13 ± 3.40E−12	6.53E−14 ± 5.41E−14	4.29E−14 ± 3.96E−14
f_{cec05_2}	**4.35E−31** ± 7.72E−26	1.34E+03 ± 1.14E+04	1.82E−02 ± 9.53E−02	1.26E−04 ± 2.28E−04	3.38E+03 ± 3.19E+03	2.85E−03 ± 1.52E−03	2.95E−06 ± 6.21E−06
f_{cec05_3}	9.42E+02 ± 3.45E+02	1.24E+00 ± 3.64E+01	**7.39E−14** ± 1.43E−14	8.31E+06 ± 6.43E+06	4.06E−09 ± 1.27E−08	6.73E−01 ± 2.30E+00	5.21E+05 ± 4.17E+05
f_{cec05_4}	**2.03E+00** ± 1.67E+00	1.07E+04 ± 7.25E+04	9.46E+02 ± 3.30E+03	5.48E+02 ± 4.39E+02	1.12E+04 ± 1.57E+04	6.26E+02 ± 4.91E+02	3.79E+00 ± 2.45E+00
f_{cec05_5}	**1.18E+02 ±** 2.38E+02	2.74E+04 ± 3.12E+04	2.21E+02 ± 3.27E+02	6.58E+03 ± 5.16E+03	1.86E+04 ± 1.26E+04	8.41E+03 ± 2.56E+03	2.98E+03 ± 6.24E+03
f_{cec05_6}	**3.97E−04** ± 2.57E−04	1.73E+03 ± 4.23E+04	2.99E+01 ± 1.68E+02	1.53E+01 ± 1.34E+01	1.04E+01 ± 5.67E+01	1.95E+01 ± 2.03E+01	1.16E+01 ± 8.28E+00
f_{cec05_7}	**1.91E−03** ± 1.37E−03	1.52E+03 ± 1.68E+03	5.63E+03 ± 2.85E+03	1.20E+00 ± 1.17E+00	5.42E+03 ± 1.43E+03	4.28E+01 ± 2.55E+01	2.96E−02 ± 3.76E−02
f_{cec05_8}	**2.03E+01** ± 2.27E−02	2.09E+01 ± 3.70E−01	2.12E+01 ± 3.60E−01	2.09E+01 ± 2.40E−01	2.10E+01 ± 1.90E−01	2.09E+01 ± 3.00E−01	2.09E+01 ± 1.80E−01
f_{cec05_9}	2.99E+00 ± 2.72E+00	3.55E+01 ± 5.88E+01	5.74E+00 ± 9.09E+00	8.36E+00 ± 3.27E+00	**2.00E−01** ± 2.59E+00	3.36E+00 ± 1.69E+00	6.57E+01 ± 2.57E+01
$f_{cec05_{10}}$	3.18E+01 ± 1.60E+01	2.21E+02 ± 8.70E+01	1.33E+02 ± 3.00E+02	2.98E+00 ± 1.43E+00	1.42E+02 ± 8.45E+01	1.52E−06 ± 2.13E−07	**0.00E+00** ± 0.00E+00
$w/t/l$	–	9/0/1	9/0/1	9/0/1	7/1/2	8/0/2	9/0/1

4.6. Comparison results on a set of 10 CEC 2005 shifted benchmarks

For the real world problems, it cannot be expected that the optima always locate in the center and have the same values for different dimensions. So, to further verify the performance of CSPSO, a set of 10 CEC 2005 shifted benchmark functions listed in Table 18 are used in this test. More detailed definitions of these benchmark functions can be found in [61].

In the section, six state-of-the-art PSO variants listed in Table 19 are used for comparison. The related parameters in the PSO variants are set according to the corresponding references.

In order to make a fair comparison, according to all compared PSO variants, the number of dimensions of all the test functions is set to $D = 30$, the population size is set to $M = 40$, and the maximum number of function evaluations in a run for all the test functions is set to $Max\text{FEs} = 3 \times 10^5$. To reduce statistical errors, each test is repeated 25 times independently.

The experimental results on the mean error values and the standard deviations are reported in Table 20. The comparison results between CSPSO and other PSO variants are summarized as "$w/t/l$" in the last row of the table and the statistical results obtained by the Friedman test and the post hoc test are listed in Tables 21 and 22.

From the results in Table 20, it is observed that CSPSO achieves better results than DNSCLPSO on eight functions, loses in two functions. For the popular algorithm CLPSO, CSPSO wins in seven functions, ties in one function and loses in two functions. Compared with CPSO-H, APSO, GOPSO and DNSPSO, CSPSO wins in nine functions, only loses in one function.

Table 21 shows the average ranks of all the seven PSO variants. The highest rank is shown in bold. As seen, the performance of the five algorithms can be sorted by average ranks into the following order: CSPSO, DNSPSO, DNSCLPSO, GOPSO, APSO, CLPSO, and CPSO-H. The highest average rank is obtained by the CSPSO algorithm. It demonstrates that CSPSO is the best one among the seven PSO variants for the 10 CEC 2005 shifted benchmark functions.

Table 21
Average ranks of CSPSO and other PSOs on 10 CEC 2005 benchmark functions achieved by the Friedman tests. The highest rank is shown in bold.

Algorithms	Ranks
CSPSO	**1.80**
DNSPSO [68]	3.15
DNSCLPSO [68]	3.75
GOPSO [69]	3.85
APSO [77]	4.70
CLPSO [39]	4.80
CPSO-H [66]	5.95
Statistic	23.052
p-value	0.001

Table 22
Adjusted *p*-values achieved by the Holland and Finer procedures between CSPSO and six other PSO variants.

CSPSO vs.	Holland	Finer
CPSO-H [66]	**0**	**0**
CLPSO [39]	**0**	**0**
APSO [77]	**0.00399**	**0.00199**
GOPSO [69]	**0.03849**	**0.01944**
DNSCLPSO [68]	**0.04547**	**0.02753**
DNSPSO [68]	**0.044**	**0.044**

With seven compared PSO algorithms, the statistic is distributed according to the *F* distribution with six degrees of freedom. The critical value for $\alpha = 0.05$ is 12.592, so the null-hypothesis is rejected. The *p*-value computed by using the *F* distribution is 0.001, so the Friedman test strongly suggests the existence of significant differences among the algorithms considered.

Table 22 confirms the significant improvement of CSPSO over other PSO variants. According to the adjusted *p*-values, the post hoc test successfully detects the statistical difference between CSPSO and all of the compared algorithms. This experimental results suggest that the better results achieved by CSPSO are statistically significant and are not obtained by chance.

5. Conclusions

In this paper, a novel crisscross search particle swarm optimizer called CSPSO has been developed. Compared with other PSO variants, CSPSO has several distinctive features as follows. First, the swarm in CSPSO is directly represented by a population of *pbest*s, which are renewed by the modified PSO search as well as the crisscross search in sequence at each generation. As the crisscross search could be feeding *pbest*s of high quality to the modified PSO, it accelerates the swarm to converge to the global optimum. Second, As an evolutionary catalytic agent, the crisscross search is implemented by a dual search mechanism including horizontal crossover and vertical crossover, which reproduce a population of moderation solutions at each generation by performing different crossover operations in opposite directions. In particular, the horizontal crossover search improves PSO's global search ability greatly. Unlike the horizontal crossover search, the vertical crossover search is used to keep swarm diversity and is also an effective means of avoiding the local optimum problem. Third, neither the new positions updated by the modified PSO search nor the moderation solutions generated by the crisscross search can directly enter the subsequent search phase. They must compete with the parent population stemming from the previous search phase. Only those that outperform their parent counterparts can survive and then replace the corresponding parent particles with new *pbests*. Such competitive mechanism makes CSPSO always maintain a population of *pbests*, thus speeding up the convergence rate *T*. Extensive experimental results in this work indicate that the CSPSO has superior searching performance, compared with most of the other state-of-the-art PSO variants.

Further research remains to be conducted on incorporating our crisscross search strategy into other swarm intelligence algorithms, in which, the premature convergence problem in the later period of the optimization is a ubiquitous phenomena. It would be interesting to know how the crisscross search will work in other population-based evolutionary algorithms. Besides, we are preparing to apply our CSPSO algorithm to solving more complex real-world engineering problems.

Acknowledgment

This research work is supported by the National Natural Science Fund Project of China (51307025).

References

[1] M. Abramowitz, Handbook of Mathematical Functions: With Formulas, Graphs and Mathematical Tables, Dover Publications, 1974.
[2] R.K. Agrawal, N.G. Bawane, Multiobjective PSO based adaption of neural network topology for pixel classification in satellite imagery, Appl. Soft Comput. J. 28 (2015) 217–225.
[3] R. Ahila, V. Sadasivam, K. Manimala, An integrated PSO for parameter determination and feature selection of ELM and its application in classification of power system disturbances, Appl. Soft Comput. J. 32 (2015) 23–37.
[4] P.S. Andrews, An investigation into mutation operators for particle swarm optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, 2006, pp. 1044–1051.
[5] P.J. Angeline, Using selection to improve particle swarm optimization, in: Proceedings of IEEE International Conference on Computer, 1998, pp. 84–89.
[6] H. Bevrani, F. Habibi, P. Babahajyani, M. Watanabe, Y. Mitani, Intelligent frequency control in an AC microgrid: online PSO-based fuzzy tuning approach, IEEE Trans. Smart Grid. 3 (2012) 1935–1944.
[7] K. Chan, T.S. Dillon, E. Chang, An intelligent particle swarm optimization for short-term traffic flow forecasting using on-road sensor systems, IEEE Trans. Ind. Electron. 60 (2013) 4714–4725.
[8] H. Chen, Y. Zhu, K. Hu, T. Ku, Global optimization based on hierarchical convolutions model, in: Proceedings of IEEE Congress on Evolutionary Computation, 2008, pp. 1497–1504.
[9] P. Chen, Two-level hierarchical approach to unit commitment using expert system and elite PSO, IEEE Trans. Power Syst 27 (2012) 780–789.
[10] X. Chen, Y. Li, A modified PSO structure resulting in high exploration ability with convergence guaranteed, IEEE Trans. Syst. Man Cybern. 37 (2007) 1271–1289.
[11] G. Chen, L. Liu, P. Song, Y. Du, Chaotic improved PSO-based multi-objective optimization for minimization of power losses and L index in power systems, Energy Convers. Manage. 86 (2014) 548–560.
[12] W. Chen, J. Zhang, Y. Lin, N. Chen, Z. Zhan, H. Chung, Y. Li, Y. Shi, Particle swarm optimization with an aging leader and challengers, IEEE Trans. Evol. Comput. 17 (2013) 241–258.
[13] L. Chuang, S. Tsai, C. Yang, Catfish particle swarm optimization, in: Proceedings of IEEE Swarm Intelligence Symposium, 2008, pp. 1–5.
[14] L. Chuang, S. Tsai, C. Yang, Chaotic catfish particle swarm optimization for solving global numerical optimization problems, Int. J. Appl. Math. Comput. Sci. 217 (2011) 6900–6916.
[15] M. Clerc, J. Kennedy, The particle swarm—explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (2002) 58–73.
[16] G. Das, P.K. Pattnaik, S.K. Padhy, Artificial neural network trained by particle swarm optimization for non-linear channel equalization, Expert Syst. Appl. 41 (2014) 3491–3496.
[17] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[18] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. 1 (2011) 3–18.
[19] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995, pp. 39–43.
[20] R. Eberhart, Y. Shi, Particle swarm optimization: developments, applications and resources, in: Proceedings of IEEE Congress on Evolutionary Computation, 2001, pp. 81–86.
[21] W. Elloumi, H. El Abed, A. Abraham, A.M. Alimi, A comparative study of the improvement of performance using a PSO modified by ACO applied to TSP, Appl. Soft Comput. 25 (2014) 234–241.
[22] A. Engelbrecht, Heterogeneous particle swarm optimization, in: Proceedings of the Seventh International Conference on Swarm Intelligence, 2010, pp. 191–202.
[23] M. Eslami, H. Shareef, M. Khajehzadeh, A. Mohamed, A survey of the state of the art in particle swarm optimization, Res. J. Appl. Sci. Eng. Technol. 4 (2012) 1181–1197.
[24] N.V. George, G. Panda, A particle-swarm-optimization-based decentralized nonlinear active noise control system, IEEE Trans. Instrum. Meas. 61 (2012) 3378–3386.
[25] M. Gomez-Gonzalez, A. López, F. Jurado, Corrigendum to "Optimization of distributed generation systems using a new discrete PSO and OPF", Electr. Power Syst. Res. 121 (2015) 379 [Electr. Power Syst. Res. (2012) 84 (1) 174–180].
[26] H. Higashi, H. Iba, Particle swarm optimization with Gaussian mutation, in: Proceedings of IEEE Swarm Intelligence Symposium, 2003, pp. 72–79.
[27] F. Hu, F. Wu, Diploid hybrid particle swarm optimization with differential evolution for open vehicle routing problem, in: Proceedings of the Eighth World Congress on Automatic Control and Artificial Intelligence, 2010, pp. 2692–2697.
[28] X. Hu, R.C. Eberhart, Multiobjective optimization using dynamic neighborhood particle swarm optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, pp. 1677–1681.
[29] K. Ishaque, Z. Salam, M. Amjad, S. Mekhilef, An improved particle swarm optimization (PSO)-based MPPT for PV with reduced steady-state oscillation, IEEE Trans. Power Electron. 27 (2012) 3627–3638.
[30] N. Jin, Y. Rahmat-Samii, Hybrid real-binary particle swarm optimization (HPSO) in engineering electromagnetics, IEEE Trans. Ant. Prop. 58 (2010) 3786–3794.
[31] C.F. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, IEEE Trans. Syst. Man Cybern. B Cybern. 34 (2004) 997–1006.
[32] S. Kachroudi, M. Grossard, N. Abroug, Predictive driving guidance of full electric vehicles using particle swarm optimization, IEEE Trans. Vehicle Technol. 61 (2012) 3909–3919.
[33] J. Kennedy, Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance, in: Proceedings of IEEE Congress on Evolutionary Computation, 1999, pp. 1931–1938.
[34] J. Kennedy, Bare bones particle swarms, in: Proceedings of IEEE Swarm Intelligence Symposium, 2003, pp. 80–87.
[35] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of International Conference on Neural Networks, 1995, pp. 1942–1948.
[36] J. Kennedy, R. Mendes, Population structure and particle swarm performance, in: Proceedings of IEEE Congress on Evolutionary Computation, vol. 2, 2002, pp. 1671–1676.
[37] J. Kennedy, R. Mendes, Neighborhood topologies in fully-in-formed and best-of-neighborhood particle swarms, in: Proceedings of the 2003 IEEE International Workshop on Soft Computing in Industrial Applications, 2003, pp. 45–50.
[38] R. Kuo, L. Lin, Application of a hybrid of genetic algorithm and particle swarm optimization algorithm for order clustering, Decis. Support Syst. 49 (2010) 451–462.
[39] J. Liang, A. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particles swarm optimization for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (2006) 281–295.
[40] J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer, in: Proceedings of IEEE Swarm Intelligence Symposium, 2005, pp. 124–129.
[41] L.W. Hong, M.I.N. Ashidi, Two-layer particle swarm optimization with intelligent division of labor, Eng. Appl. Artif. Intel. 26 (2013) 2327–2348.
[42] Y. Liu, Z. Qin, Z. Shi, J. Lu, Center particle swarm optimization, Neurocomputing 70 (2007) 672–679.
[43] Z. Liu, W. Sun, J. Zeng, A new short-term load forecasting method of power system based on EEMD and SS-PSO, Neural Comput. Appl. 24 (2014) 973–983.
[44] M. Lovbjerg, T.K. Rasmussen, T. Krink, Hybrid particle swarm optimizer with breeding and subpopulations, in: Proceedings of International Conference on Genetic and Evolutionary, 2001, pp. 469–476.

[45] R. Martínez-Soto, O. Castillo, L.T. Aguilar, Type-1 and Type-2 fuzzy logic controller design using a hybrid PSO-GA optimization method, Inf. Sci. 285 (2014) 35–49.
[46] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Trans. Evol. Comput. 8 (2004) 204–210.
[47] A. Meng, Y. Chen, H. Yin, S. Chen, Crisscross optimization algorithm and its application, Knowl. Based Syst. 67 (2014) 218–229.
[48] A.S. Mohais, C. Ward, C. Posthoff, Randomized directed neighborhoods with edge migration in particle swarm optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, 2004, pp. 548–555.
[49] M.A. Montes de Oca, T. Stutzle, M. Birattari, M. Dorigo, Frankenstein's PSO: a composite particle swarm optimization algorithm, IEEE Trans. Evol. Comput. 13 (2009) 1120–1132.
[50] V.F. Nepomuceno, P. Andries, A. Engelbrecht, Self-adaptive heterogeneous PSO for real-parameter optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, 2013, pp. 20–23.
[51] M. Ramezani, M. Haghifam, C. Singh, H. Seifi, M.P. Moghaddam, Determination of capacity benefit margin in multiarea power systems using particle swarm optimization, IEEE Trans. Power Syst. 2 (2009) 631–641.
[52] A. Ratnaweera, S. Halgamuge, H. Waston, Self-organizing hierarchical particle optimizer with time-varying acceleration coefficients, IEEE Trans. Evol. Comput. 8 (2004) 240–255.
[53] N. Sa-ngawong, I. Ngamroo, Intelligent photovoltaic farms for robust frequency stabilization in multi-area interconnected power system based on PSO-based optimal Sugeno fuzzy logic control, Renew. Energy 74 (2015) 555–567.
[54] Y. Shang, Y. Qiu, A note on the extended Rosenbrock function, Evol. Comput. 14 (2006) 119–126.
[55] P.S. Shelokar, P. Siarry, V.K. Jayaraman, B.D. Kulkarni, Particle swarm and ant colony algorithms hybridized for improved continuous optimization, Int. J. Appl. Math. Comput. 188 (2007) 129–142.
[56] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings of IEEE Congress on Evolutionary Computation, 1998, pp. 69–73.
[57] Y. Shi, R.C. Eberhart, Parameters selections in particle swarm optimization, in: Proceedings of IEEE International Conference on Evolutionary Programming, 1998, pp. 591–600.
[58] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, 1999, pp. 1945–1950.
[59] Y. Shi, R.C. Eberhart, Fuzzy adaptive particle swarm optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, 2001, pp. 101–106.
[60] P.H. Silva, R.M.S. Cruz, A.G.D. Assuncao, Blending PSO and ANN for optimal design of FSS filters with Koch Island patch elements, IEEE Trans. Magn. 46 (2010) 3010–3013.
[61] P.N. Suganthan, Particle swarm optimizer with neighborhood operator, in: Proceedings of IEEE Congress on Evolutionary Computation, 1999, pp. 1958–1962.
[62] P.N. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, in: Proceedings of Technical Report, Nanyang Technological University, Singapore, May 2005 AND KanGAL Report, 2005.
[63] Y. Tang, Z. Wang, J. Fang, Feedback learning particle swarm optimization, Appl. Soft Comput. 11 (2011) 4713–4725.
[64] N.R. Tayebi, F.M. Nejad, M. Mola, Comparison between GA and PSO in analyzing pavement management activities, J. Transp. Eng. 40 (2014) 99–104.
[65] F. Valdez, P. Melin, O. Castillo, Modular neural networks architecture optimization with a new nature inspired method using a fuzzy combination of particle swarm optimization and genetic algorithms, Inf. Sci. 270 (2014) 143–153.
[66] F. van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. 8 (2004) 225–239.
[67] R. Wai, J. Lee, K. Chuang, Real-time PID control strategy for maglev transportation system via particle swarm optimization, IEEE Trans. Ind. Electron. 58 (2011) 629–646.
[68] H. Wang, H. Sun, C. Li, S. Rahnamayan, J.S. Pan, Diversity enhanced particle swarm optimization with neighborhood search, Inf. Sci. 223 (2013) 119–135.
[69] H. Wang, Z.J. Wu, S. Rahnamayan, Y. Liu, M. Ventresca, Enhancing particle swarm optimization using generalized opposition-based learning, Inf. Sci. 181 (2011) 4699–4714.
[70] J. Wang, F. Yang, Optimal capacity allocation of standalone wind/solar/battery hybrid power system based on improved particle swarm optimisation algorithm, IET Renew. Power Generat. 7 (2013) 443–448.
[71] M. Xi, J. Sun, W. Xu, An improved quantum-behaved particle swarm optimization algorithm with weighted mean best position, Appl. Math. Comput. 205 (2008) 751–759.
[72] X. Xie, P. Wu, Research on the optimal combination of ACO parameters based on PSO, in: Proceedings of the Second International Conference on Networking and Digital Society, 2010, pp. 94–97.
[73] B. Xin, J. Chen, J. Zhang, H. Fang, Z. Peng, Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: a review and taxonomy, IEEE Trans. Syst. Man Cybern. C Appl. Rev. 42 (2012) 744–767.
[74] T. Xiong, Y. Bao, Z. Hu, R. Chiong, Forecasting interval time series using a fully complex-valued RBF neural network with DPSO and PSO algorithms, Inf. Sci. 305 (2015) 77–92.
[75] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (1999) 82–102.
[76] S.H. Yeung, W.S Chan, K.T. Ng, K.F. Man, Computational optimization algorithms for antennas and RF/microwave circuit designs: an overview, IEEE Trans. Ind. Inf. 8 (2012) 216–227.
[77] Z. Zhan, J. Zhang, Y. Li, H. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst. Man Cybern. B Cybern. 39 (2009) 1362–1381.
[78] Z. Zhan, J. Zhang, Y. Li, Y. Shi, Orthogonal learning particle swarm optimization, IEEE Trans. Evol. Comput. 15 (2011) 832–847.
[79] Y. Zhong, Y. Xiang, Y. Jiang, Z.J. Hong, J. Shao, A hybrid dynamic multi-swarm PSO algorithm with Nelder–Mead simplex search method, Comput. Inf. Syst. 9 (2013) 7741–7748.
[80] D. Zhou, X. Gao, G. Liu, C. Mei, D. Jiang, Y. Liu, Randomization in particle swarm optimization for global search ability, Expert Syst. Appl. 38 (2011) 15356–15364.
[81] T. Zhuang, Q. Li, Guo, X. Wang, A two-stage particle swarm optimizer, in: Proceedings of IEEE Congress on Computational Intelligence, 2008, pp. 557–563.

Further Reading

R. Thangaraj, M. Pant, A. Abraham, P. Bouvry, Particle swarm optimization: hybridization perspectives and experimental illustrations, Appl. Math. Comput. 217 (2011) 5208–5226.