Information Sciences 291 (2015) 43–60

Contents lists available at ScienceDirect

Information Sciences

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

A social learning particle swarm optimization algorithm for scalable optimization

Ran Cheng [a], Yaochu Jin [a,b,*]

[a] Department of Computing, University of Surrey, Guildford, Surrey GU2 7XH, United Kingdom
[b] College of Information Sciences and Technology, Donghua University, Shanghai 201620, China

ARTICLE INFO

Article history:
Received 19 October 2013
Received in revised form 5 August 2014
Accepted 16 August 2014
Available online 2 September 2014

Keywords:
Social learning
Particle swarm optimization
Large-scale optimization
Computational efficiency
Scalability

ABSTRACT

Social learning plays an important role in behavior learning among social animals. In contrast to individual (asocial) learning, social learning has the advantage of allowing individuals to learn behaviors from others without incurring the costs of individual trials-and-errors. This paper introduces social learning mechanisms into particle swarm optimization (PSO) to develop a social learning PSO (SL-PSO). Unlike classical PSO variants where the particles are updated based on historical information, including the best solution found by the whole swarm (global best) and the best solution found by each particle (personal best), each particle in the proposed SL-PSO learns from any better particles (termed demonstrators) in the current swarm. In addition, to ease the burden of parameter settings, the proposed SL-PSO adopts a dimension-dependent parameter control method. The proposed SL-PSO is first compared with five representative PSO variants on 40 low-dimensional test functions, including shifted and rotated test functions. The scalability of the proposed SL-PSO is further tested by comparing it with five state-of-the-art algorithms for large-scale optimization on seven high-dimensional (100-D, 500-D, and 1000-D) benchmark functions. Our comparative results show that SL-PSO performs well on low-dimensional problems and is promising for solving large-scale problems as well.

1. Introduction

Particle Swarm Optimization (PSO) is one powerful and widely used swarm intelligence paradigm introduced by Kennedy and Eberhart in 1995 for solving optimization problems [43]. In PSO, it is assumed that each particle is able to memorize the best position found in history, i.e., the best position that has ever been found by the whole swarm, termed *global best*, and the best position that has ever been found by each particle, known as *personal best*. To find the global optimum of the optimization problem, the particles learn from the personal best and global best positions. Specifically, the learning mechanisms in the canonical PSO can be summarized as follows:

$$V_i(t+1) = \omega V_i(t) + c_1 R_1(t)(pbest_i(t) - X_i(t)) + c_2 R_2(t)(gbest(t) - X_i(t)), \quad (1)$$

$$X_i(t+1) = X_i(t) + V_i(t+1), \quad (2)$$

* Corresponding author at: Department of Computing, University of Surrey, Guildford, Surrey GU2 7XH, United Kingdom. Tel.: +44 1483686037; fax: +44 1483686051.
E-mail address: yaochu.jin@surrey.ac.uk (Y. Jin).

http://dx.doi.org/10.1016/j.ins.2014.08.039
0020-0255/

where t is the generation number, $V_i(t)$ and $X_i(t)$ represent the velocity and position of the i-th particle, respectively; ω is termed inertia weight [71], c_1 and c_2 are the acceleration coefficients [20], $R_1(t)$ and $R_2(t)$ are two vectors randomly generated within $[0,1]^n$, with n being the dimension of the search space; $pbest_i(t)$ and $gbest(t)$ denote the personal best of the i-th particle and the global best of the swarm, respectively.

Although PSO has witnessed rapid developments over the past decades and has been successfully applied to a number of applications, e.g., water distribution network design [56], resource allocation [24], task assignment [30], maximum power point tracker for photovoltaic system [36], optimal control [67], DNA sequence compression [100] and reconstruction of gene regulatory network [60], image processing [69], text clustering [6] and many others [75,34,52], its performance is still unsatisfying when the optimization problem has a large number of local optima or when the optimization problem is high-dimensional and non-separable [87]. The poor performance of PSO can largely be attributed to its weak robustness to various problem structures [84]. In order to enhance the search performance of the canonical PSO, numerous PSO variants have been proposed, which can be roughly divided into the following five categories.

The first category adopts an adaptive control strategy of the parameters in PSO. As shown in (1), ω, c_1 and c_2 are the three control parameters in PSO. ω, called the inertia weight, was introduced as a constant in [72] and an adaptive parameter in [73]. Another important modification of ω is the introduction of fuzzy inference [74]. Adaptation strategies for tuning the acceleration coefficients, c_1 and c_2, have also been proposed [64,11]. Apart from adapting ω, c_1 and c_2, additional parameters to help regulate these three parameters have also been introduced [95,33].

Hybrid PSO algorithms, which can be categorized into the second category of PSO variants, combine PSO with other search strategies, such as genetic algorithms [65,39,80], differential evolution [39], and ant colony optimization [70]. Other specific search operators have also been incorporated into PSO, including mutation operators [29,51,61], local search operators [83,86], and some nature-inspired operators as well, such as the niching PSO [5], culture-based PSO [13] and the aging theory inspired PSO (ALC-PSO) [8].

The third category of PSO variants introduces new topological structures in neighborhood control to enhance swarm diversity, thereby alleviating premature convergence [76,41]. Several topological structures have been proposed [44], e.g., the ring topology and the von Neumann topology. In [53], a fully informed PSO (FIPS) was developed, where the update of each particle is based on the historical best positions of several neighbors instead of *gbest* or *pbest*. Another representative example is the comprehensive learning PSO (CLPSO) introduced in [49], where particles update each decision variable by learning from different historical personal best positions. More recently, a distance-based locally informed particle swarm optimizer has been proposed to tackle multi-modal problems [63].

Multi-swarm PSO belongs to the fourth category. The main idea behind multi-swarm PSO variants is to enhance swarm diversity by exchanging information between different swarms. For example, the dynamic multi-swarm PSO (DMS-PSO) [48] was proposed with a dynamically changing neighborhood structure. In [81], a cooperative multi-swarm PSO (CPSO) is proposed to solve large-scale optimization problems by dividing the decision variables into several sub-components. Similarly, the recently proposed cooperative coevolving PSO (CCPSO2) for large-scale optimization adopts the multi-swarm paradigm as well [46]. Additional multi-swarm PSO variants have been reported in [2,92,19,10].

The fifth category of PSO variants is designed to perform effective search using limited computational cost and memory usage. Unlike the first four categories, these PSO variants aim to simplify PSO to enhance its robustness by taking into account the constraints of the device on which the PSO is running, rather than introducing additional mechanisms to enhance the search performance for specific problems. Most PSO variants in this category are based on a probabilistic model. The earliest model-based PSO was proposed by Kennedy [42], called the Bare Bones PSO (BBPSO), which uses Gaussian distributions instead of the velocity equation to update the particle positions. In [99], a simplified intelligent single particle optimization (ISPO) was proposed to explore the search space using a single particle instead of a swarm, and some further discussions on the ISPO can also be found in a recently published literature [35]. In [77], a fitness estimation strategy has been introduced based on the relationship between the positions of the particles to reduce the required number of fitness evaluations. Most recently, a competitive swarm optimizer based on pairwise competition between particles was reported [9], which has been shown to perform effectively for large scale optimization problems and needs much less memory compared to the standard PSO.

PSO algorithms are often believed to be different from other population-based meta-heuristics in that PSO does not adopt an explicit selection mechanism. However, as pointed out in [7,58], most PSO variants can also be captured by a generic framework that is valid for almost all population-based meta-heuristics including evolutionary algorithms such as genetic algorithms (GAs) and swarm intelligence algorithms such as PSO. A slight difference lies in the fact that most PSO algorithms apply selection implicitly, e.g. by updating the personal best solutions and the global best solution only. While evolutionary algorithms such as GAs perform selection by comparing a list of solutions for multiple times, PSO compares solutions pairwise. As suggested in [9,58], pairwise comparison in selection can not only reduce memory usage, but also make it easier to adopt a compact encoding.

This paper proposes a new PSO variant by further modifying the implicit selection mechanism in PSO, where neither a global best solution nor personal best solutions will be stored. Instead, the swarm is sorted according to the fitness of the particles and all particles except for the best one will be updated by learning from any particle whose fitness is better. In a sense, selection in the proposed algorithm is implicitly performed in the sorting process.

Learning and imitating the behaviors of better individuals in the population, which is known as social learning, can widely be observed among social animals. Social learning, different from individual (asocial) learning, has the advantage of allowing

individuals to learn behaviors from others without incurring the costs of individual trial and error, which is able to accelerate learning rates [45], especially when the target (behavior) to learn is complex. More specifically, individual learning is a process of trial and error, whilst social learning is to take advantage of mechanisms such as imitation, enhancement and conditioning [27].

Due to the appealing properties of social learning, it is quite natural to apply the social learning mechanisms to population-based stochastic optimization. In 2011, Oca and Stutzle proposed an incremental social learning framework, which consists of a growing population of agents that learn socially when they become part of the main group and learn individually when they are already part of it [57]. To the best of our knowledge, this is the earliest attempt to explicitly apply social learning mechanism to optimization. However, the proposed framework is very generic and does not involve any concrete social learning mechanism such as *imitation*, which plays an important role in natural social learning. Moreover, the proposed framework is not able to work independent of an existing optimizer [15], e.g., a PSO variant.

In this paper, we introduce social learning mechanisms into PSO, resulting in a substantially different PSO variant, which is termed social learning PSO (SL-PSO). Unlike most PSO variants, SL-PSO is performed on a sorted swarm. Instead of learning from the historical best positions, the particles learn from any better particles (demonstrators) in the current swarm. To ease the burden of parameter setting, a *dimension-dependent* parameter control strategy has been suggested in the proposed SL-PSO to enhance its robustness to the search dimensionality (the number of decision variables) of the problem to be optimized.

The rest of this paper is organized as follows. Section 2 describes the proposed SL-PSO in detail, together with an analysis of time complexity and a proof of convergence. In Section 3, the performance of the SL-PSO is examined by comparing its performance with a few widely reported optimization algorithms on low-dimensional and large-scale optimization problems. Finally, conclusions are drawn in Section 4.

2. A social learning particle swarm optimizer

In the following, a brief account of the sociological background for the proposed SL-PSO will be given. Then, a detailed description of SL-PSO will be provided together with an analysis of the computational complexity and a convergence proof.

2.1. Sociological background

In 1921, some British birds were first seen to open milk bottles in the small town of Swaythling. In the following 25 years, such observations had been continually reported from numerous other sites spreading all over the Great Britain and even some other areas in the European continent. This is the first evidence of *social learning* [22], where the birds are believed to learn to open milk bottles by observations and interactions with other birds, instead of learning by themselves [28].

During the past decades, various mechanisms have been proposed and discussed in social learning theory, e.g., stimulus enhancement and local enhancement [27], observational conditioning [54], contagion [93] and social facilitation [3]. Among these mechanisms, the most interesting social learning mechanism is *imitation*, which is considered to be distinctive from other social learning mechanisms [94], because imitation, which operates across a whole community, could lead to population-level similarities of behavior such as *culture* or *tradition* [85]. Such population-level similarities may imply *convergence* of a dynamic system, thus providing its essential applicability in an evolutionary algorithm.

In [14], the authors had a detailed discussion about different definitions of imitation, among which Mitchell's definition is considered to be the most applicable one across animals and machines [55], in which imitation is considered to be a procedure to make a similar (but not exactly the same) copy of a model. In [85], imitation is described as a procedure of an imitator copying *part of* a behavior from a demonstrator via observation.

In the following, we present a few new learning mechanisms inspired from social learning to replace the updating rules in the canonical PSO.

2.2. Algorithm description

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

$$\begin{aligned} &\min f = f(X), \\ &\text{s.t.} \quad X \in \mathcal{X}, \end{aligned} \tag{3}$$

where $\mathcal{X} \subset \mathbb{R}^n$ is the feasible solution set, n denotes the dimensionality of the search space, i.e., the number of decision variables, which are the behaviors to learn in the context of social learning.

2.2.1. The overall framework

Like the classical PSO, the proposed SL-PSO initializes a swarm $P(t)$ containing m particles, where m is the swarm size and t is the generation index. For each particle i in $P(t)$, it holds a randomly initialized decision vector (behavior vector) $X_i(t)$, which represents a candidate solution to the optimization problem described in (3). As a reward feedback from the environment, every particle will be assigned with a fitness value calculated from the objective function $f(X)$. The swarm is then sorted according to an increasing order of the particles' fitness values. Consequently, each particle (except for the one with

the best fitness value) will correct its behaviors by learning from those particles (demonstrators) which have better fitness values. The general framework illustrating the above procedure is shown in Fig. 1.

2.2.2. Swarm sorting and behavior learning

It can be seen from Fig. 1 that, apart from fitness evaluations, the most important components in SL-PSO are *swarm sorting* and *behavior learning*. For an easy description of the behavior learning mechanisms, the swarm is first sorted according to the particles' fitness values. Then, each particle i (an *imitator*), except for the best one, will learn from its corresponding demonstrators. Note that in each generation, a particle could serve as an demonstrator for different imitators more than once. In a sorted swarm, for any imitator (particle i, where $1 \leqslant i < m$), its demonstrators can be any particle k that satisfies $i < k \leqslant m$, refer to Fig. 2. For example, for particle 1, particles 2, 3, . . ., m can be its demonstrators, while for particle (m-1), only particle m can be its demonstrator. As a result, particle 1 (the worst one) can never be a demonstrator and particle m (the best one) will never be an imitator. That is, the best particle in the current swarm will not be updated.

Inspired by social learning mechanism, an imitator will learn the behaviors of different demonstrators [13] in the following manner:

$$X_{i,j}(t+1) = \begin{cases} X_{i,j}(t) + \Delta X_{i,j}(t+1), & \text{if } p_i(t) \leqslant P_i^L, \\ X_{i,j}(t), & \text{otherwise}, \end{cases} \tag{4}$$

where $X_{i,j}(t)$ is the j-th dimension of particle i's behavior vector in generation t, with $i \in \{1, 2, 3, \ldots, m\}$ and $j \in \{1, 2, 3, \ldots, n\}$, $\Delta X_{i,j}(t+1)$ is the behavior correction. Taking into account the fact that in a society, the motivation to learn

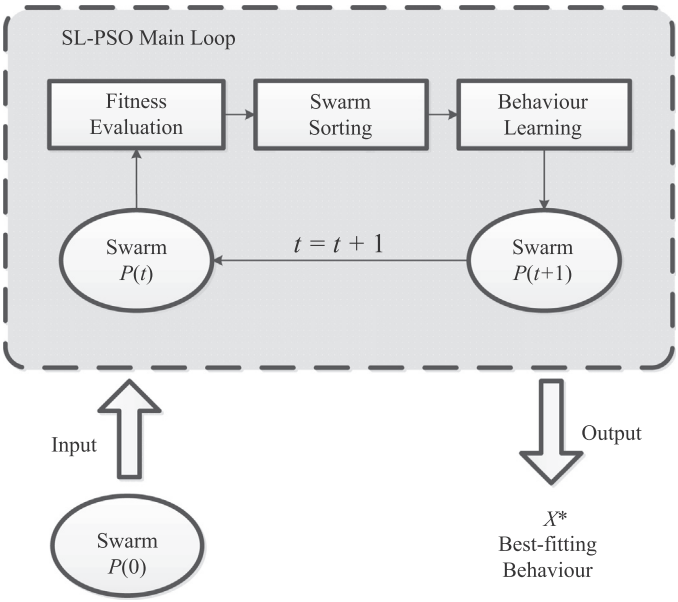

Fig. 1. Main components of the proposed SL-PSO.

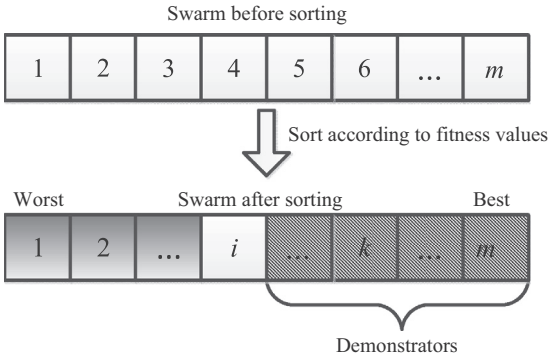

Fig. 2. Swarm sorting and behavior learning in SL-PSO. At first, the swarm is sorted according to the fitness values; then each particle (except for the best one) learns from its demonstrators which have better fitness values.

from better individuals may vary from individual to individual (typically, better individuals are less willing to learn from others), we define a *learning probability* P_i^L for each particle i. The precise definition of the learning probability will be discussed later on. As a result, particle i will learn (correct its behavior vector) only if a randomly generated probability p_i satisfies $0 \leqslant p_i(t) \leqslant P_i^L \leqslant 1$. In detail, $\Delta X_{ij}(t+1)$ is generated as follows:

$$\Delta X_{ij}(t+1) = r_1(t) \cdot \Delta X_{ij}(t) + r_2(t) \cdot I_{ij}(t) + r_3(t) \cdot \epsilon \cdot C_{ij}(t), \tag{5}$$

with

$$\begin{cases} I_{ij}(t) = X_{kj}(t) - X_{ij}(t), \\ C_{ij}(t) = \overline{X}_j(t) - X_{ij}(t). \end{cases} \tag{6}$$

In the above updating mechanisms inspired from social learning, the behavior correction $\Delta X_{ij}(t+1)$ consists of three components. The first component $\Delta X_{ij}(t)$ is the same as the *inertia component* in the canonical PSO, while the other two components are different. In the second component, instead of learning from *pbest* as done in the canonical PSO, particle i learns from any of its demonstrators. Specifically, the j-th element in the behavior vector of particle i, $X_{ij}(t)$ imitates $X_{kj}(t)$, which is the j-th element in the behavior vector of particle k (demonstrator of particle i). Note that $i < k \leqslant m$, and k is generated independently for each element j, refer to Fig. 2. Consequently, particle i may learn from different demonstrators in the current swarm. Since this component is inspired from the imitation behavior in natural social learning, it is denoted as *imitation component*. Likewise, particle i does not learn from *gbest* either; instead, it learns from the collective behavior of the whole swarm, i.e., the mean behavior of all particles in the current swarm, denoted by $\overline{X}_j(t)$, where $\overline{X}_j(t) = \frac{\sum_{i=1}^{m} X_i^j}{m}$. Since this component induces a swarm-level conformity [93], it is denoted as the *social influence component*, and correspondingly, the control parameter ϵ is denoted as the *social influence factor*. For simplicity, the three control parameters in classical PSO (ω, c_1 and c_2) have been replaced with three random coefficients $r_1(t), r_2(t)$ and $r_3(t)$, which will be randomly generated within $[0, 1]$ once the updating strategy is performed.

2.3. Dimension-dependent parameter control

In the proposed SL-PSO, there are three parameters that need to be defined, i.e., the swarm size m, the learning probability P_i^L, and the social influence factor ϵ. As discussed in Section 1, the robustness of a PSO algorithm can be enhanced by adopting adaptive parameter control, such that it could be applied to different problems without laborious parameter fine-tunings. It has been found that the performance of most PSO variants is more sensitive to the search dimensionality of the optimization problem than other widely used evolutionary algorithms [82], we suggest a dimension-dependent parameter control strategy for the proposed SL-PSO as a guideline. It should be pointed out that this parameter control strategy has been suggested based on pilot empirical studies, nevertheless, its effectiveness has been verified on 47 test functions, refer to Section 3.

The first parameter to be determined is the swarm size m. We recommend that the swarm size m be determined as a function of the search dimensionality in the following form:

$$m = M + \left\lfloor \frac{n}{10} \right\rfloor, \tag{7}$$

where M is the base swarm size for the SL-PSO to work properly. It has been empirically shown that a small swarm size is usually sufficient for uni-modal optimization problems while a bigger swarm size is required for multi-modal optimization problems for more intensive exploration [25,1]. As in real-world applications it is difficult to know in advance whether a problem is uni-modal or multi-modal, we suggest $M = 100$ in this work.

The idea for setting the learning probability P_i^L is also inspired from natural social learning. As previously mentioned, within a swarm, the better the fitness a particle has, the less likely the particle will learn from others. Meanwhile, the more complex the behavior is, the less likely a particle tends to successfully learn the behavior. Generally speaking, the performance of most meta-heuristic algorithms degrades as the search dimensionality of the optimization problem increases, in particular when there exist strong correlations between the decision variables. Thus, we assume that the higher the search dimensionality is, the more difficult it is to solve the problem, and therefore, the less likely a particle is willing to learn from others. Based on such an assumption, we recommend a inversely proportional relationship between the learning probability and the problem dimensionality.

Based on the discussions above, the following learning probability has been adopted:

$$P_i^L = \left(1 - \frac{i-1}{m}\right)^{\alpha \cdot \log(\lceil \frac{n}{M} \rceil)}, \tag{8}$$

where the radix component $\left(1 - \frac{i-1}{m}\right)$ indicates that the learning probability is inversely proportional to the particle index i in a sorted swarm, meaning that the higher the fitness of a particle is, the lower the learning probability will be. Meanwhile, the exponent component $\alpha \cdot \log\left(\lceil \frac{n}{M} \rceil\right)$ indicates that the learning probability is inversely proportional to the search dimensionality, such that a better swarm diversity would be maintained for large-scale problems due to the reduced learning rate, and the $\alpha \cdot \log(\cdot)$ function is used to smoothen the influence of $\frac{n}{M}$. Empirically, we recommend the coefficient $\alpha < 1$, and in this work, $\alpha = 0.5$ has been used.

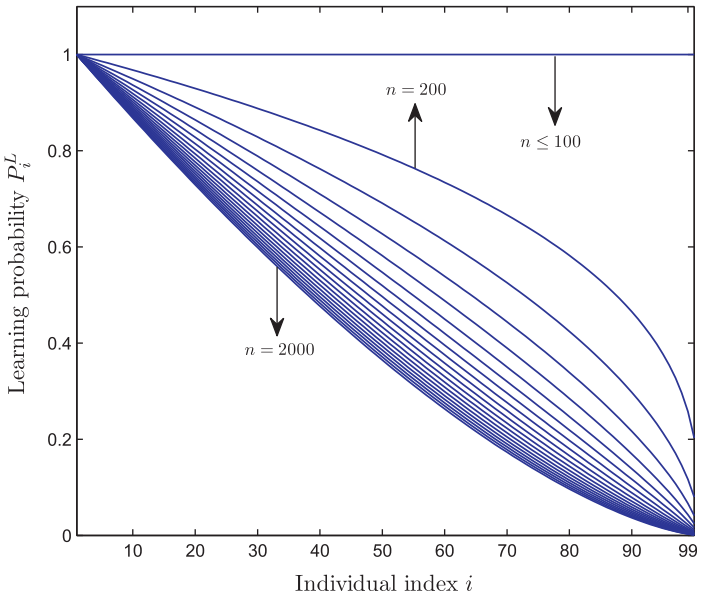

Fig. 3. The learning probability curves for different search dimensionality varying from $n \leqslant 100$ to $n = 2000$. Here, a fixed swarm size $m = 100$ is used and the swarm is already sorted according to behavior fitness values.

In order to obtain a more intuitive understanding of the relationship between the learning probability P_i^L, swarm size m and search dimensionality n, a number of curves showing the relationship between the learning probability and the search dimensionality varying from $n \leqslant 100$ to $n = 2000$ are plotted in Fig. 3. It can be seen that, when the dimensionality is not large, e.g., $n \leqslant 100$, the learning probability keeps constant at 1 for all particles. By contrast, when the dimensionality becomes larger, the learning probability decreases as the fitness value increases (a higher index in the sorted swarm) or as the dimensionality (n) becomes higher. It could also be noticed that, under the influence of $\alpha \cdot \log(\cdot)$ function, the probability curves decrease sharply at the very beginning whilst more gently with the increase of n.

The last parameter that remains to be specified is the social influence factor ϵ. Typically, the difficulty in convergence is proportional to the search dimensionality, because the convergence of a whole swarm requires the convergence of each dimension in each particle's behavior vector. Based on this observation, the social influence factor ϵ is specified as follows:

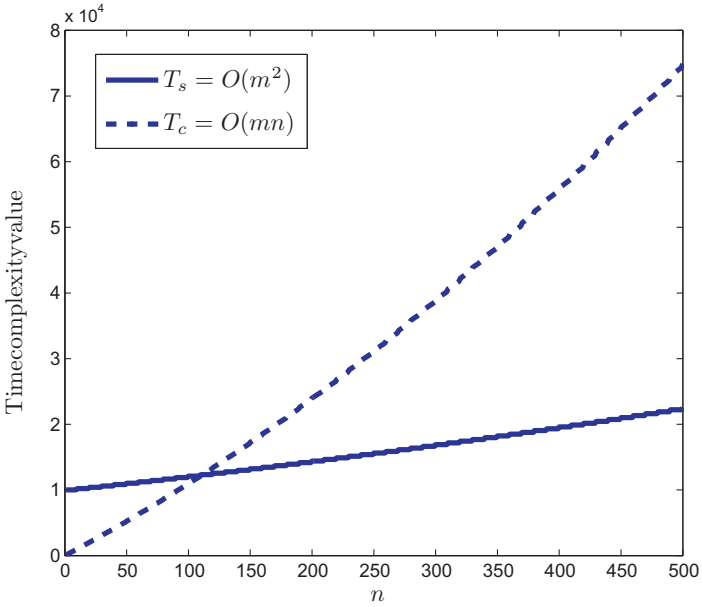

Fig. 4. The time complexity for swarm sorting and behavior learning, denoted by T_s and T_c, respectively, when the search dimensionality n varies from 1 to 500, given the swarm size $m = 100$.

$$\epsilon = \beta \times \frac{n}{M}, \tag{9}$$

which means that ϵ is proportional to the problem dimension. Since ϵ controls the influence of the swarm-level mean behavior, premature convergence to the mean behavior (rather than the best behavior) could occur if the value of this parameter is set too big. Therefore, a small value of $\beta = 0.01$ is used in this work.

In summary, the swarm size m, the learning probability P_i^L, and the social influence factor ϵ are set up based on the search dimensionality with an aim to achieve a good balance between convergence and diversity. Typical values for M, α and β are recommended and our results show that the proposed SL-PSO is able to perform robustly on problems of various search dimensionality scales without additional fine tuning of the parameters.

Algorithm 1. The pseudocode of the proposed SL-PSO. t is the generation number. Unless otherwise specified, the *termination condition* is the maximum number of fitness evaluations.

/*initialization*/
$t = 0$;
$M = 100, \alpha = 0.5, \beta = 0.01$;
$m = M + \lfloor \frac{n}{10} \rfloor$; // refer to (7)
$\epsilon = \beta \times \frac{n}{M}$; // refer to (22)
for $i = 1$ to m **do**
  randomly initialize behavior X_i in $P(0)$;
  $P_i^L = \left(1 - \frac{i-1}{m}\right)^{\alpha \cdot \log(\lceil \frac{n}{M} \rceil)}$; // refer to (5)
end for
/*main loop, refer to Fig. 1*/
while *termination condition* is not satisfied **do**
  /*fitness evaluation*/
  for $i = 1$ to m **do**
    $F_i = \mathrm{f}(X_i(t))$; //$f(X)$ is the objective function
  end for
  update best solution X^*;
  /*behavior learning, refer to Fig. 2*/
  sort $P(t)$ according to behavior fitness values in F;
  for $i = 1$ to $m - 1$ **do**
    /*correct behavior $X_i(t)$ according to (4)–(6)*/
    $p_i(t) = randr(0, 1)$; //$randr(a, b)$ randomly generates an real value between a and b
    if $p_i(t) \leqslant P_i^L$ **then**
      for $j = 1$ to n **do**
        $k = randi(i + 1, m)$; //$randi(A, B)$ randomly generates an integer between A and B
        $\Delta X_{ij}(t + 1) = \Delta X_{ij}(t) + I_{ij}(t) + \epsilon \times C_{ij}(t)$;
        $X_{ij}(t + 1) = X_{ij}(t) + \Delta X_{ij}(t + 1)$;
      end for
    end if
  end for
  $t = t + 1$;
end while
output X^* as the final solution.

2.4. Computational complexity

Analysis of the computational complexity of the proposed SL-PSO is fairly straightforward thanks to its simplicity. As shown in Fig. 1 and Algorithm 1, the computational complexity of SL-PSO depends on three parts: fitness evaluations, swarm sorting and behavior learning. In real-world applications, the time cost of fitness evaluations is problem-dependent [38,50], which is beyond the scope of this work. Therefore, the analysis of time complexity below focuses on swarm sorting and behavior learning.

Since the swarm sorting in SL-PSO is a typical sequence sorting procedure, the worst computational complexity to sort the swarm consisting of m-particles is:

$$T_s = O(m^2), \tag{10}$$

which can be attained by various sorting algorithms such as quick sort [31]. According to (7), m is a constant for a search dimensionality $n \leqslant 100$ and gradually increases with the search dimensionality n when $n > 100$. For instance, for $n = 500$,

according to (7), $m = M + 50$. Therefore, the time consumption for swarm sorting will not increase dramatically as the search dimensionality n increases. Instead, it can be seen as a constant determined by the basic population size M.

Behavior learning is an essential and common process for updating particle behaviors, which is also similar to other learning mechanisms in various swarm intelligence algorithms [18,40,43,59]. The time complexity of behavior learning is indispensable in a population-based stochastic search algorithm. Given an m-particle swarm and an n-dimensional optimization problem, the time complexity can be obtained as follows:

$$T_c = O(mn). \tag{11}$$

Further to the above analysis, some empirical results on the influence of the search dimensionality n on the time complexity for swarm sorting and behavior learning, T_s, T_c, respectively, are presented in Fig. 4. In Section 4, comparative experimental results on computational time will confirm the low time complexity of SL-PSO compared to other PSO variants.

2.5. Convergence proof

Similar to most theoretical convergence analysis of PSO [12,79,21], a deterministic implementation of the proposed SL-PSO is considered to theoretically analyze its convergence property. It should also be pointed out that the proof does not guarantee a convergence to the global optimum.

Without loss of generality, the convergence of the whole swarm can be more specifically regarded as the convergence of every dimension in any particle behavior vector. In other words, the convergence can be satisfied if and only if there is no more change in any dimension of the behavior vector of all particles. Theoretically, there should exist an *equilibrium* to induce such convergence [68]. To start with, we consider the update of the j-th $(1 \leqslant j \leqslant n)$ dimension in the behavior vector of particle $i (1 \leqslant i \leqslant m), X_{i,j}(t)$. Once its learning probability satisfies $p_i(t) \leqslant P_i^L, X_{i,j}(t)$ will be corrected as follows:

$$X_{i,j}(t+1) = X_{i,j}(t) + \Delta X_{i,j}(t+1). \tag{12}$$

If we substitute (6) into (5) and replace all random parameters with their expected value, the following expression can be obtained:

$$\Delta X_{i,j}(t+1) = \frac{1}{2}\Delta X_{i,j}(t) + \frac{1}{2}(X_{k,j}(t) - X_{i,j}(t)) + \frac{1}{2}\epsilon \times (\overline{X}_j(t) - X_{i,j}(t)), \tag{13}$$

where $\frac{1}{2}$ is the expected value of r_1, r_2 and r_3.

In this way, the convergence proof of the proposed SL-PSO can be reduced to a convergence proof of the dynamic system described by (12) and (13).

Theorem 1. *The dynamic system described by (12) and (13) converges to an equilibrium.*

Proof. Let

$$\begin{aligned} \theta &= \frac{1+\epsilon}{2}, \\ p &= \frac{1}{1+\epsilon}X_{k,j}(t) + \frac{\epsilon}{1+\epsilon}\overline{X}_{i,j}(t), \end{aligned} \tag{14}$$

then (12) and (13) can be rewritten together into:

$$\begin{aligned} \Delta X_{i,j}(t+1) &= \frac{1}{2}\Delta X_{i,j}(t) + \theta(p - X_{i,j}(t)), \\ X_{i,j}(t+1) &= X_{i,j}(t) + \Delta X_{i,j}(t+1). \end{aligned} \tag{15}$$

The search dynamics described in (15) can be seen as a dynamical system, and the convergence analysis of the system can be conducted by using the well established theories on stability analysis in dynamical systems. To this end, we rewrite system (15) into the following form:

$$y(t+1) = Ay(t) + Bp, \tag{16}$$

where

$$y(t) = \begin{bmatrix} \Delta X_{i,j}(t) \\ X_{i,j}(t) \end{bmatrix}, \quad A = \begin{bmatrix} \frac{1}{2} & -\theta \\ \frac{1}{2} & 1-\theta \end{bmatrix}, \quad B = \begin{bmatrix} \theta \\ \theta \end{bmatrix}, \tag{17}$$

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

If there exists an *equilibrium* y^* that satisfies $y^*(t+1) = y^*(t)$ for any t, it can be calculated from (16) and (17):

$$y^* = [\,0 \quad p\,], \tag{18}$$

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

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

$$\lambda^2 - \left(\frac{3}{2} - \theta\right)\lambda + \frac{1}{2} = 0, \tag{19}$$

where the eigenvalues are:

$$\begin{cases} \lambda_1 = \frac{3}{4} - \frac{\theta}{2} + \frac{\sqrt{\left(\frac{3}{2}-\theta\right)^2 - 2}}{2}, \\ \lambda_2 = \frac{3}{4} - \frac{\theta}{2} - \frac{\sqrt{\left(\frac{3}{2}-\theta\right)^2 - 2}}{2}. \end{cases} \tag{20}$$

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

$$\theta > 0, \tag{21}$$

where if θ is substituted by ϵ using (14), the condition for convergence on ϵ is:

$$\epsilon > -1. \tag{22}$$

Further, if ϵ is substituted by n using (9), the condition becomes:

$$n > -\frac{M}{\beta}. \tag{23}$$

Therefore, since $n > 0 > -\frac{M}{\beta}$ is always satisfied, the convergence condition of the system can be guaranteed. □

3. Experimental studies

The proposed SL-PSO has been tested on 47 functions, including 12 widely used basic test functions [91,4,97,96,8] (f_1 to f_{12}, refer to Table 1), 28 functions taken from the whole test suite for CEC'13 special session on real-parameter optimization (f_{13} to f_{40}, refer to [47]), and 7 functions taken from the whole test suite for CEC'08 special session on large scale global optimization (f_{41} to f_{47}, refer to [78]), where f_{13} to f_{47} are all shifted or rotated functions.

In order to verify the scalability of the proposed SL-PSO, test functions of different dimensions have been used in the experiments. Firstly, the proposed SL-PSO is tested on twelve 30-D functions (f_1 to f_{12}) and 28 50-D functions (f_{13} to f_{40}) in comparison with five widely reported PSO variants proposed for low-dimensional optimization. Afterwards, SL-PSO is further tested on (f_{41} to f_{47} by setting the search dimensionality to 100-D, 500-D and 1000-D, respectively. For comparison, five representative meta-heuristic algorithms specifically designed for large-scale optimization problems have been tested.

All experimental results are obtained from 30 independent runs. The experiments have been conducted on a PC with an Intel Core i3-2328 2.2 GHz CPU and Microsoft Windows 7 Enterprize SP1 64-bit operating system. In the experiments on low-dimensional test problems, the proposed SL-PSO and the compared PSO variants are implemented in Matlab 2010a. For the experiments on high-dimensional (large-scale) optimization problems, SL-PSO is implemented in C with Microsoft Visual Studio 2010 Enterprize for better computational efficiency.

It should also be noted that, thanks to the dimension-dependent parameter control strategy introduced in Section 2.3, no additional parameter settings for SL-PSO are needed. The termination condition, which is also the baseline for comparison, is the maximum number of fitness evaluations (FEs).

3.1. Performance on low-dimensional test problems

In the experiments on low-dimensional optimization problems, five representative PSO variants have been chosen to compare with the proposed SL-PSO, including the global version PSO (GPSO) [73], the local version PSO (LPSO) [44], the fully informed PSO (FIPS) [53], the dynamic multi-swarm PSO (DMS-PSO) [48] and the comprehensive learning PSO (CLPSO) [49]. The parameter settings for these PSO variants are summarized in Table 2.

Firstly, the proposed SL-PSO and the five PSO variants are tested on the 12 basic functions (f_1 to f_{12}), among which f_1 to f_5 are uni-modal functions, f_6 is a step function whose minimum is non-continuous, and f_7 to f_{12} are multi-modal functions. The dimension of these functions is set to 30 and the termination condition of each algorithm is met when a maximum number of 2×10^5 fitness evaluations is exhausted.

The results yielded by the proposed SL-PSO and the PSO variants on f_1 to f_{12} are summarized in Table 3. In each row of the table, the mean values averaged over 30 independent runs are listed in the first line, and the standard deviations are listed in the second line. Two-tailed t-test is performed with a significance level $\alpha = 0.05$. In the table, if SL-PSO statistically

Table 1
Low-dimensional test functions used in the experiments.

Name	Function	Search range
Sphere	$f_1(X) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$
Schwefel 2.22	$f_2(X) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} x_i$	$[-10, 10]^n$
Schwefel 1.2	$f_3(X) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^n$
Schwefel 2.21	$f_4(X) = \max \lvert x_i, i \leqslant i \leqslant n \rvert$	$[-100, 100]^n$
Rosenbrock	$f_5(X) = \sum_{i=1}^{n-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$	$[-30, 30]^n$
Step	$f_6(X) = \sum_{i=1}^{n} \lfloor x_i + 0.5 \rfloor^2$	$[-100, 100]^n$
Schwefel	$f_7(X) = 418.9829 \cdot n + \sum_{i=1}^{n} -x_i \cdot \sin(\sqrt{\lvert x_i \rvert})$	$[-500, 500]^n$
Rastrigin	$f_8(X) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5.12, 5.12]^n$
Ackley	$f_9(X) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e$	$[-32, 32]^n$
Griewank	$f_{10}(X) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^n$
Penalized 1	$f_{11}(X) = \frac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$, $y = 1 + \frac{1}{4}(x_i + 1)$	$[-50, 50]^n$
Penalized 2	$f_{12}(X) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	$[-50, 50]^n$

In f_{11} and f_{12}, $u(x_j, a, k, m) = \begin{cases} k(x_j - a)^m, & x_j > a \\ 0, & -a \leqslant x_j \leqslant a \\ k(-x_j - a)^m, & x_j < -a \end{cases}$.

All functions are scalable to the search dimensionality denoted by n.
The global optimum is 0 for all functions.

Table 2
Parameter settings for the compared PSO variants.

Algorithm	Parameter settings
GPSO	$\omega = 0.90.4, c_1 = c_2 = 2.0$
LPSO	$\omega = 0.90.4, c_1 = c_2 = 2.0$
FIPS	$\chi = 0.729, \sum c_i = 4.1$
DMS-PSO	$\omega = 0.729, c_1 = c_2 = 1.49445, m = 3, R = 15$
CLPSO	$\omega = 0.90.7, c_1 = c_2 = 1.49445$

significantly outperforms another algorithm, a bold '+' is labeled in front of the corresponding result obtained by this algorithm; if there is no significance between SL-PSO and the compared algorithm, a '=' is labeled and if SL-PSO is outperformed, a '−' is labeled. At the bottom of each table that contains the statistical results, a summary of total number of '+', '=' and '−' is listed. In addition, the best results obtained for each function are highlighted in bold. Note that all results are optimization errors, i.e., the difference between the finally obtained fitness values and the global optimum values. The same presentation format is used in all tables listing the comparative results.

In general, SL-PSO has shown the best performance on 9 out of 12 functions (f_1 to f_4, f_6 and f_9 to f_{12}), including four uni-modal functions and five multi-modal functions. Although DMS-PSO and CLPSO have both shown outstanding performance on f_7 and f_8, SL-PSO has outperformed them on almost all the rest functions, which indicates that SL-PSO has better overall performance on this test suite. On the one hand, SL-PSO has maintained the merit of fast convergence feature of PSO, which can be confirmed by its performance on the uni-modal functions like f_1; on the other hand, its performance on multi-modal functions has been enhanced, which can be demonstrated by its performance on f_9 to f_{12}. In order to further justify the above conclusion, the convergence profiles of one typical uni-modal function (f_1) and one typical multi-modal function (f_{11}) are potted in Fig. 5. It can be seen that, the convergence speed of SL-PSO ranks first on both functions. On the contrary, the convergence speed of DMS-PSO and CLPSO are slower, even in comparison with the GPSO and LPSO, which are two basic PSO variants. In other words, the high performance of DMS-PSO and CLPSO on some multi-modal problems (e.g. f_7 and f_8)

Table 3
Optimization errors on 12 basic test functions.

30-D	SL-PSO	GPSO		LPSO		FIPS		DMS-PSO		CLPSO	
f_1	**4.24E−90**	+	1.25E−61	=	8.48E−35	+	6.20E−70	=	3.30E−14	+	4.76E−19
	5.26E−90		2.82E−61		2.85E−34		1.44E−69		1.27E−13		1.92E−19
f_2	**1.50E−46**	+	7.33E+00	=	6.67E−01	+	1.13E−38	+	8.48E−11	+	7.54E−12
	5.34E−47		1.39E+01		2.58E+00		5.70E−39		1.84E−10		2.50E−12
f_3	**4.66E−07**	+	4.22E+03	+	3.65E−01	+	1.21E+00	+	9.79E+01	+	1.13E+03
	2.48E−07		5.08E+03		3.83E−01		6.59E−01		7.31E+01		2.89E+02
f_4	**1.17E−24**	+	8.49E−07	+	4.42E−05	+	2.37E+00	+	1.90E+00	+	4.31E+00
	8.37E−25		1.01E−06		2.32E−05		1.17E+00		7.85E−01		6.84E−01
f_5	2.15E+01	+	6.05E+03	+	5.18E+01	+	3.53E+01	+	5.60E+01	−	**9.28E+00**
	3.41E+00		2.32E+04		3.68E+01		2.71E+01		3.28E+01		**1.03E+01**
f_6	**0.00E+00**	=	0.00E+00	=	0.00E+00	=	0.00E+00	+	5.33E−01	=	0.00E+00
	0.00E+00		0.00E+00		0.00E+00		0.00E+00		9.15E−01		0.00E+00
f_7	1.50E+03	+	5.62E+03	+	3.07E+03	+	2.98E+03	−	5.74E−08	−	**6.06E−13**
	9.10E+01		2.19E+03		7.80E+02		7.87E+02		6.02E−10		**8.88E−13**
f_8	1.55E+01	+	4.65E+01	+	5.02E+01	+	3.86E+01	−	**2.70E−13**	−	5.83E−09
	3.19E+00		2.55E+01		2.25E+01		1.04E+01		**8.41E−13**		5.02E−09
f_9	**5.51E−15**	+	1.36E−14	+	7.67E+00	+	6.69E−15	+	6.11E−09	+	2.99E−10
	1.59E−15		4.34E−15		9.79E+00		1.83E−15		1.89E−08		9.47E−11
f_{10}	**0.00E+00**	+	1.21E−02	+	2.46E−03	+	2.07E−13	+	1.76E−02	+	8.40E−12
	0.00E+00		1.58E−02		6.64E−03		5.03E−13		2.56E−02		1.45E−11
f_{11}	**1.57E−32**	=	6.91E−03	=	1.57E−32	=	1.57E−32	=	9.32E−15	+	3.61E−20
	0.00E+00		2.68E−02		2.83E−48		2.83E−48		3.61E−14		1.87E−20
f_{12}	**1.35E−32**	=	7.32E−04	=	7.32E−04	=	1.35E−32	=	1.47E−03	+	3.31E−19
	0.00E+00		2.84E−03		2.84E−03		2.83E−48		3.87E−03		8.67E−20
+/=/−		8/4/0		6/6/0		9/3/0		7/3/2		8/1/3	

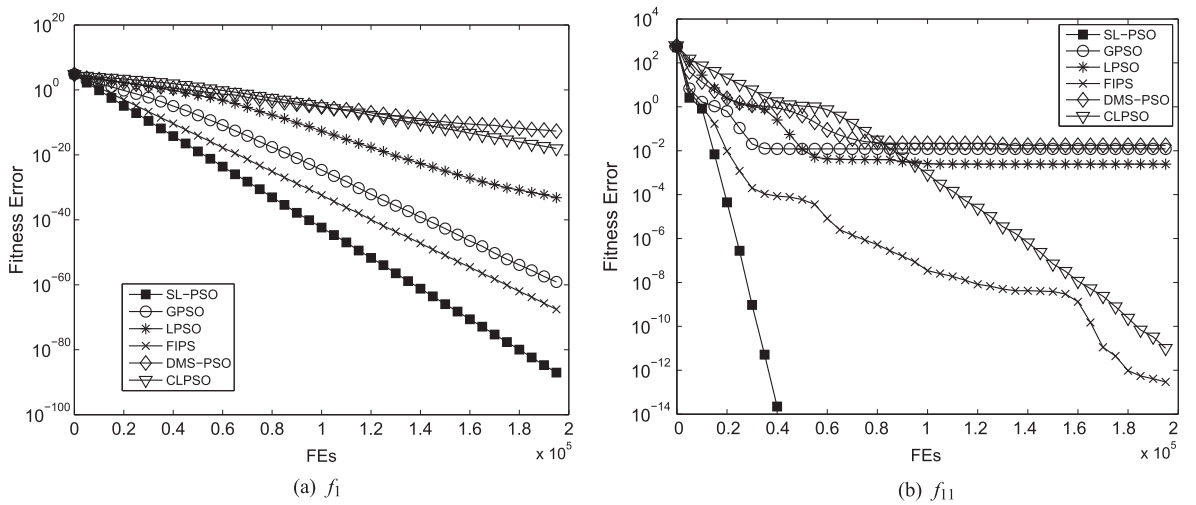

Fig. 5. The convergence profiles.

might have been achieved by sacrificing the fast convergence feature of PSO. In addition, although GPSO, LPSO and FIPS have shown fast convergence on f_1, they suffer premature convergence on multi-modal functions like f_{11}.

In order to further verify the robustness of the proposed SL-PSO, the whole function suite (denoted from f_{13} to f_{40}) proposed in the CEC'13 special session [47] has been taken for comparative studies in this work. The test suite consists of five uni-modal functions (f_{13} to f_{17}), 15 multi-modal functions (f_{18} to f_{32}) and eight composition functions (f_{33} to f_{40}). All the functions are shifted or rotated to increase their complexity. In this group of tests, the dimension of the functions is set to 50 and the termination condition of each algorithm is a maximum number of 2.5×10^5 fitness evaluations.

Table 4
Optimization errors on CEC'13 functions.

50-D	SL-PSO	GPSO		LPSO		FIPS		DMS-PSO		CLPSO	
f_{13}	**2.05E−13**	+	5.29E+03	+	2.34E+02	+	4.55E−13	+	8.27E−06	+	5.91E−13
	7.19E−14		5.37E+03		3.20E+02		0.00E+00		1.12E−05		1.25E−13
f_{14}	**2.22E+06**	+	8.34E+07	+	5.12E+07	+	2.71E+07	+	1.11E+07	+	6.55E+07
	5.90E+05		9.54E+07		5.60E+07		5.01E+06		5.91E+06		9.86E+06
f_{15}	**1.31E+07**	+	9.81E+10	+	4.96E+08	+	6.61E+08	+	3.58E+08	+	5.92E+09
	2.13E+07		5.91E+10		2.03E+08		2.82E+08		1.74E+08		1.69E+09
f_{16}	3.82E+04	−	**1.58E+04**	−	2.44E+04	+	4.26E+04	−	3.21E+04	+	6.47E+04
	3.11E+03		**1.39E+04**		4.82E+03		7.63E+03		2.26E+03		1.06E+04
f_{17}	**1.82E−13**	+	1.59E+03	+	4.88E+02	+	5.68E−13	+	5.32E−04	+	5.26E−09
	6.23E−14		9.92E+02		3.52E+02		8.14E−14		5.69E−04		2.03E−09
f_{18}	**4.52E+01**	+	4.88E+02	+	1.14E+02	=	4.55E+01	+	4.74E+01	+	4.79E+01
	2.33E+00		6.51E+02		9.30E+01		1.13E+00		7.84E−01		4.30E−01
f_{19}	**7.49E+00**	+	1.63E+02	+	1.83E+02	+	9.38E+01	+	5.63E+01	+	1.14E+02
	1.51E+00		4.06E+01		3.41E+01		6.94E+00		7.67E+00		1.06E+01
f_{20}	2.12E+01	=	2.12E+01	−	**2.11E+01**	=	2.12E+01	=	2.12E+01	=	2.12E+01
	3.44E−02		9.10E−03		**4.59E−02**		2.51E−02		2.86E−02		5.49E−02
f_{21}	**1.82E+01**	+	4.43E+01	+	5.03E+01	+	5.93E+01	+	4.72E+01	+	5.71E+01
	1.75E+00		2.67E+00		4.91E+00		2.20E+00		3.48E+00		2.62E+00
f_{22}	**2.41E−01**	+	1.45E+03	+	3.43E+02	+	2.97E−01	+	6.99E+00	+	2.95E+01
	1.06E−01		1.29E+03		2.52E+02		7.31E−02		3.59E+00		8.24E+00
f_{23}	2.65E+01	+	1.56E+02	+	1.24E+02	+	1.32E+02	−	5.92E+00	−	**7.97E−05**
	5.87E+00		8.03E+01		5.87E+01		1.91E+01		3.87E+00		**6.60E−05**
f_{24}	3.39E+02	=	3.73E+02	−	1.89E+02	+	4.09E+02	−	**1.26E+02**	=	3.31E+02
	2.11E+01		1.89E+02		5.03E+01		1.02E+01		**6.61E+01**		3.14E+01
f_{25}	3.43E+02	+	5.87E+02	=	3.25E+02	+	4.18E+02	−	**2.36E+02**	+	4.07E+02
	2.45E+01		1.49E+02		5.66E+01		1.89E+01		**3.66E+01**		3.49E+01
f_{26}	1.08E+03	+	2.59E+03	+	6.53E+03	+	1.07E+04	−	**1.87E+01**	−	1.44E+02
	3.95E+02		1.03E+03		1.89E+03		7.91E+02		**1.06E+01**		2.65E+01
f_{27}	1.23E+04	−	**7.76E+03**	−	8.11E+03	+	1.41E+04	−	9.22E+03	−	1.04E+04
	3.16E+03		**5.53E+02**		1.02E+03		3.95E+02		2.75E+03		8.86E+02
f_{28}	3.33E+00	−	2.09E+00	−	2.52E+00	=	3.50E+00	−	**2.01E+00**	−	2.82E+00
	3.27E−01		3.48E−01		3.98E−01		3.54E−01		**8.40E−01**		6.00E−01
f_{29}	3.70E+02	=	3.46E+02	−	1.18E+02	−	3.56E+02	−	6.47E+01	−	**6.28E+01**
	1.41E+01		1.94E+02		2.21E+01		3.25E+01		3.58E+00		**1.42E+00**
f_{30}	3.97E+02	−	3.45E+02	−	2.57E+02	+	4.44E+02	−	**2.08E+02**	+	4.52E+02
	9.63E+00		7.95E+01		7.24E+01		1.59E+01		**9.37E+01**		1.86E+01
f_{31}	9.18E+00	+	4.36E+04	+	6.44E+01	+	2.94E+01	−	4.15E+00	−	**3.92E+00**
	5.24E+00		5.87E+04		8.74E+01		1.72E+00		1.12E+00		**6.39E−01**
f_{32}	2.24E+01	=	2.22E+01	−	**2.12E+01**	−	2.17E+01	+	2.35E+01	+	2.38E+01
	3.47E−01		1.06E+00		**5.65E−01**		4.98E−01		1.01E+00		5.50E−01
f_{33}	7.60E+02	=	9.33E+02	+	1.19E+03	−	**3.62E+02**	=	7.88E+02	−	4.91E+02
	4.06E+02		1.27E+03		1.83E+02		**2.73E+02**		4.22E+02		2.19E+02
f_{34}	1.14E+03	+	4.25E+03	+	6.79E+03	+	1.05E+04	−	**5.54E+01**	−	6.07E+02
	3.05E+02		5.37E+02		7.83E+02		1.04E+03		**4.76E+01**		1.45E+02
f_{35}	1.10E+04	=	1.06E+04	=	9.56E+03	+	1.46E+04	−	**7.95E+03**	=	1.15E+04
	4.27E+03		1.34E+03		1.94E+03		2.37E+02		**1.06E+03**		8.47E+02
f_{36}	**2.46E+02**	+	3.37E+02	+	3.62E+02	+	3.37E+02	+	2.71E+02	+	3.55E+02
	8.35E+00		2.57E+01		4.02E+00		9.04E+00		1.50E+01		3.28E+00
f_{37}	**2.94E+02**	+	4.81E+02	+	3.80E+02	+	3.73E+02	+	3.29E+02	+	3.94E+02
	8.23E+00		1.00E+01		8.25E+00		1.57E+01		1.18E+01		8.18E+00
f_{38}	3.35E+02	+	4.17E+02	+	4.45E+02	−	2.50E+02	+	3.88E+02	−	**2.07E+02**
	7.91E+00		1.74E+01		4.01E+00		1.07E+02		2.62E+01		**1.30E+00**
f_{39}	**7.29E+02**	+	1.68E+03	+	1.81E+03	+	1.74E+03	+	1.19E+03	+	1.87E+03
	8.12E+01		9.36E+01		4.35E+01		1.30E+02		8.34E+01		6.00E+01
f_{40}	**4.00E+02**	+	4.33E+03	+	3.17E+03	+	4.00E+02	+	1.68E+03	+	4.00E+02
	2.05E−13		9.57E+02		1.59E+03		6.00E−03		1.75E+03		1.22E−02
+/=/−		18/6/4		18/2/8		21/3/4		14/2/12		16/3/9	

Table 5
The details of functions on which DMS-PSO and CLPSO have yielded best performance.

Function number	Detail
f_{23}	**Rastrigin**'s Function
f_{24}	Rotated **Rastrigin**'s Function
f_{25}	Non-Continuous Rotated **Rastrigin**'s Function
f_{26}	**Schwefel**'s Function
f_{27}	Rotated **Schwefel**'s Function
f_{28}	Rotated Katsuura Function
f_{29}	Lunacek Bi_**Rastrigin** Function
f_{30}	Rotated Lunacek Bi_**Rastrigin** Function
f_{31}	Expanded Griewank's plus Rosenbrock's Function
f_{34}	Composition Function of **Schwefel**'s Function (f_{26})
f_{35}	Composition Function of Rotated **Schwefel**'s Function (f_{27})

As the results shown in Table 4, the proposed SL-PSO still shows the best overall performance compared with the five PSO variants. Among the 28 functions, SL-PSO has yielded the best results on 12 of them (f_{13} to f_{15},f_{17} to f_{19},f_{21},f_{22},f_{36},f_{37},f_{39} and f_{40}). In comparison, GPSO and LPSO performs best on f_{16},f_{27} and f_{20},f_{32}, respectively; FIPS performs best on f_{33}; DMS-PSO performs best on seven functions (f_{24} to f_{26},f_{28},f_{30},f_{34} and f_{35}); and CLPSO performs best on f_{24},f_{29} and f_{31}. It can be seen that DMS-PSO and CLPSO still have shown the most competitive performance in comparison with the rest three PSO variants, because they have yielded most of the best results from f_{23} to f_{35}. For a clearer observation on those functions where DMS-PSO and CLPSO perform best, Table 5 has listed the function details. In can be seen that among the 11 functions, nine of them are the variants of Schwefel's function (f_7) and Rastrigin's function (f_8), which are precisely the two test functions on which DMS-PSO and CLPSO have shown best performance in the previous set of experiments on the 12 basic functions, refer to Tables 1 and 3. It seems that DMS-PSO and CLPSO can perform particularly well on these two functions, on which SL-PSO performs relatively poorly. However, except for the variants of these two functions, SL-PSO shows significantly better overall performance on the rest of the test functions. More importantly, the more competitive performance of SL-PSO has been achieved without sacrificing the computational efficiency of the classical PSO. This is clearly supported by the averaged computational times of the six compared algorithms summarized in Fig. 6. From the figure, we can see that SL-PSO needs the lowest computational time among the six PSO variants, even lower than the GPSO and LPSO, which are two basic PSO variants. These results confirm the time complexity analysis in Section 2.4.

3.2. Performance on optimization of high-dimensional problems

In the optimization of low-dimensional (30-D and 50-D) problems, SL-PSO has shown robust performance on 40 benchmark functions in comparison with five representative PSO variants, in addition to its high computational efficiency. However, we are keen to further investigate its performance on large-scale (high-dimensional) optimization problems,

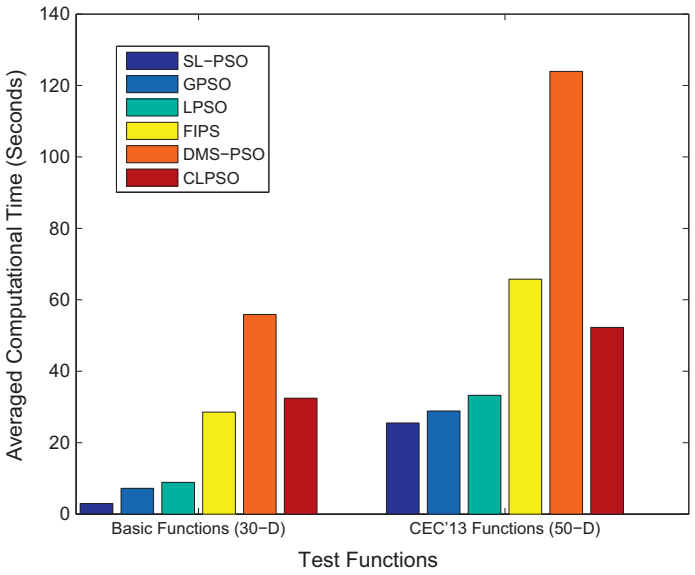

Fig. 6. The averaged computational time.

Table 6
Optimization errors on CEC'08 functions.

	SL-PSO		CCPSO2		MLCC		sep-CMA-ES		DMS-L-PSO		EPUS-PSO	
100-D												
f_{41}	1.09E−27	+	7.73E−14	+	6.82E−14	+	9.02E−15	−	**0.00E+00**	+	7.47E−01	
	3.50E−28		3.23E−14		2.32E−14		5.53E−15		**0.00E+00**		1.70E−01	
f_{42}	**9.45E−06**	+	6.08E+00	+	2.53E+01	+	2.31E+01	+	3.65E+00	+	1.86E+01	
	4.97E−06		7.83E+00		8.73E+00		1.39E+01		7.30E−01		2.26E+00	
f_{43}	5.74E+02	=	4.23E+02	−	1.50E+02	−	**4.31E+00**	−	2.83E+02	+	4.99E+03	
	1.67E+02		8.65E+02		5.72E+01		**1.26E+01**		9.40E+02		5.35E+03	
f_{44}	7.46E+01	−	3.98E−02	−	**4.39E−13**	+	2.78E+02	+	1.83E+02	+	4.71E+02	
	1.21E+01		1.99E−01		**9.21E−14**		3.43E+01		2.16E+01		5.94E+01	
f_{45}	**0.00E+00**	+	3.45E−03	+	3.41E−14	=	2.96E−04	=	0.00E+00	+	3.72E−01	
	0.00E+00		4.88E−03		1.16E−14		1.48E−03		0.00E+00		5.60E−02	
f_{46}	**2.10E−14**	+	1.44E−13	+	1.11E−13	+	2.12E+01	−	0.00E+00	+	2.06E+00	
	5.22E−15		3.06E−14		7.87E−15		4.02E−01		0.00E+00		4.40E−01	
f_{47}	−1.48E+03	−	−1.50E+03	−	**−1.54E+03**	+	−1.39E+03	+	−1.14E+03	+	−8.55E+02	
	1.90E+01		1.04E+01		**2.52E+00**		2.64E+01		8.48E+00		1.35E+01	
500-D												
f_{41}	7.24E−24	+	7.73E−14	+	4.30E−13	+	2.25E−14	−	**0.00E+0**	+	8.45E+01	
	2.05E−25		3.23E−14		3.31E−14		6.10E−15		**0.00E+00**		6.40E+00	
f_{42}	**3.47E+01**	+	5.79E+01	+	6.67E+01	+	2.12E+02	+	6.89E+01	+	4.35E+01	
	1.03E+00		4.21E+01		5.70E+00		1.74E+01		2.01E+00		5.51E−01	
f_{43}	6.10E+02	+	7.24E+02	+	9.25E+02	−	**2.93E+02**	+	4.67E+07	+	5.77E+04	
	1.87E+02		1.54E+02		1.73E+02		**3.59E+01**		5.87E+06		8.04E+03	
f_{44}	2.72E+03	−	3.98E−02	−	**1.79E−11**	+	2.18E+03	+	1.61E+03	+	3.49E+03	
	3.25E+02		1.99E−01		**6.31E−11**		1.51E+02		1.04E+02		1.12E+02	
f_{45}	3.33E−16	=	1.18E−03	+	2.13E−13	=	7.88E−04	−	**0.00E+00**	+	1.64E+00	
	0.00E+00		4.61E−03		2.48E−14		2.82E−03		**0.00E+00**		4.69E−02	
f_{46}	**1.46E−13**	+	5.34E−13	+	5.34E−13	+	2.15E+01	+	2.00E+00	+	6.64E+00	
	2.95E−15		8.61E−14		7.01E−14		3.10E−01		9.66E−02		4.49E−01	
f_{47}	−5.94E+03	−	−7.23E+03	−	**−7.43E+03**	−	−6.37E+03	+	−4.20E+03	+	−3.51E+03	
	1.72E+02		4.16E+01		**8.03E+00**		7.59E+01		1.29E+01		2.10E+01	
1000-D												
f_{41}	7.10E−23	+	5.18E−13	+	8.46E−13	+	7.81E−15	−	**0.00E+00**	+	5.53E+02	
	1.40E−24		9.61E−14		5.01E−14		1.52E−15		**0.00E+00**		2.86E+01	
f_{42}	8.87E+01	=	7.82E+01	+	1.09E+02	+	3.65E+02	+	9.15E+01	−	**4.66E+01**	
	5.25E+00		4.25E+01		4.75E+00		9.02E+00		7.14E−01		**4.00E−01**	
f_{43}	1.04E+03	+	1.33E+03	+	1.80E+03	=	**9.10E+02**	+	8.98E+09	+	8.37E+05	
	5.14E+01		2.63E+02		1.58E+02		**4.54E+01**		4.39E+08		1.52E+05	
f_{44}	5.89E+02	−	1.99E−01	−	**1.37E−10**	+	5.31E+03	+	3.84E+03	+	7.58E+03	
	9.26E+00		4.06E−01		**3.37E−10**		2.48E+02		1.71E+02		1.51E+02	
f_{45}	4.44E−16	+	1.18E−03	+	4.18E−13	=	3.94E−04	−	**0.00E+00**	+	5.89E+00	
	0.00E+00		3.27E−03		2.78E−14		1.97E−03		**0.00E+00**		3.91E−01	
f_{46}	**3.44E−13**	+	1.02E−12	+	1.06E−12	+	2.15E+01	+	7.76E+00	+	1.89E+01	
	5.32E−15		1.68E−13		7.68E−14		3.19E−01		8.92E−02		2.49E+00	
f_{47}	−1.30E+04	−	−1.43E+04	−	**−1.47E+04**	+	−1.25E+04	+	−7.50E+03	+	−6.62E+03	
	1.04E+02		8.27E+01		**1.51E+01**		9.36E+01		1.63E+01		3.18E+01	
+/ = /−			12/3/6		14/0/7		14/3/4		13/1/7		20/0/1	

whose search dimensionality is normally larger than 100. For this purpose, SL-PSO is tested on a large-scale optimization test set (denoted as f_{41} to f_{47}), which was proposed in the CEC'08 special session on large-scale optimization [78]. The dimension has been set to 100, 500, and 1000, respectively. Correspondingly, a maximum number of $5000 * n$ fitness evaluations is set as the termination condition, where n is the search dimensionality of the test problem.

In order to properly evaluate the performance of the proposed SL-PSO for large-scale optimization, five algorithms specifically tailored for large-scale optimization have been chosen for comparisons, including CCPSO2 [46], sep-CMA-ES [66], EPUS-PSO [32], and MLCC [90].

Among the five algorithms, CCPSO2 is the most recently proposed state-of-the-art for large-scale optimization, which belongs to the cooperative coevolution (CC) framework [89] for large-scale optimization [46], where a random grouping

Table 7
Ranking results based on the Holm-Bonferroni procedure for the compared algorithms on test functions from f_1 to f_{12}.

Rank	Algorithm	z	p	θ	h	Score
1	SL-PSO	–	–	–	–	5.58E+00
2	FIPS	−2.07E+00	1.91E−02	5.00E−02	1	4.00E+00
3	CLPSO	−2.73E+00	3.19E−03	2.50E−02	1	3.50E+00
4	LPSO	−3.49E+00	2.40E−04	1.67E−02	1	2.92E+00
5	DMS-PSO	−4.04E+00	2.71E−05	1.25E−02	1	2.50E+00
6	GPSO	−4.04E+00	2.71E−05	1.00E−02	1	2.50E+00

strategy is adopted based on the idea of *divide-and-conquer* [62]. Similarly, multi-level cooperative coevolution (MLCC) has also been proposed belonging to the CC framework [90], where a self-adaptive neighborhood search differential evolution variant (SaNSDE) is used instead of PSO as the core algorithm [88]. The sep-CMA-ES is an extension of the original CMA-ES algorithm [26], which has been shown more efficient and fairly scalable to some high-dimensional test functions up to 1000-D [66]. EPUS-PSO is another PSO variant which adjusts the swarm size according to the search results [32], and DMS-L-PSO is the DMS-PSO enhanced with a local search operator [98].

As shown from the experimental results summarized in Table 6, the proposed SL-PSO continues to exhibit the best overall performance on the 100-D, 500-D and 1000-D functions. Among all the compared algorithms, CCPSO2, MLCC and sep-CMA-ES have shown comparable performance as SL-PSO on the rest five test functions except for f_{41} and f_{45}, whilst significantly outperform the other two algorithms. However, the DMS-L-PSO is always able to find the real global optimum of f_{41} and f_{45}, regardless of the dimension, although it performs not so well on the other five test functions. The performance of the proposed SL-PSO on large-scale optimization problems is surprisingly good, because there is no specific mechanism for large-scale optimization such as the divide-and-conquer or the CC framework adopted in SL-PSO. We conjecture that the good scalability of SL-PSO might be attributed to the following two reasons. First, the social learning mechanism allows the particles to interactively and dynamically learn from each other, thereby maintaining a proper level of swarm diversity needed for handling large-scale problems. Second, the dimension dependent parameter control strategy might have contributed to the scalability. A rigorous analysis of SL-PSO's search performance on large-scale optimization remains to be investigated.

3.3. Holm-Bonferroni procedure based ranking of the compared algorithms

In this section, we further assess the performance of the compared algorithms by ranking them using the Holm-Bonferroni procedure [23,17] based on the statistical results summarized in Tables 3, 4 and 6, respectively. Here we adopt the procedure suggested in [58] for this purpose.

Given the statistical results obtained by N_a optimizers on N_p optimization problems, a score $S_i, i = 1, \ldots, N_a$ is assigned to the i-the algorithm in the following way: the algorithm showing the best performance is assigned a score of N_a, the second best a score of $N_a - 1$, and so on. This process continues until the worst-performed algorithm is assigned a score of 1. Based on the mean score averaged over all the test problems, the algorithms are ranked. Take the proposed SL-PSO as a reference algorithm and denote its score as S_0, the z values of the remaining $N_a - 1$ algorithms can be calculated as:

$$z_j = \frac{S_j - S_0}{\sqrt{\frac{N_a(Na+1)}{6N_p}}}, \tag{24}$$

where $S_j, j = 1, \ldots, N_a - 1$ denotes the score for the j-th algorithm and N_p is the number of test problems. On the basis of these z_j values, the corresponding cumulative normal distribution values p_j can be calculated, which are compared with the thresholds θ_j calculated based on the level of confidence δ (set to 0.05 in our case) as $\theta_j = \delta/(N_a - j)$. If $p_j < \theta_j$, it means that the null-hypothesis that there is no significant difference between the performance of two algorithms is rejected, denoted as $h = 1$; otherwise, the null-hypothesis is accepted, meaning there is no significant difference between the performance of the two algorithms, denoted as $h = 0$.

From the ranking results based on the Holm-Bonferroni procedure shown in Tables 7 and 8, we can observe that the proposed SL-PSO significantly outperforms the compared algorithms on all low-dimensional problems except that there is no

Table 8
Ranking results based on the Holm-Bonferroni procedure for the compared algorithm on test functions from f_{13} to f_{40}.

Rank	Algorithm	z	p	θ	h	Score
1	SL-PSO	–	–	–	–	4.50E+00
2	DMS-PSO	−7.14E−02	4.72E−01	5.00E−02	0	4.46E+00
3	CLPSO	−2.57E+00	5.06E−03	2.50E−02	1	3.21E+00
4	LPSO	−2.71E+00	3.32E−03	1.67E−02	1	3.14E+00
5	FIPS	−2.93E+00	1.70E−03	1.25E−02	1	3.04E+00
6	GPSO	−3.71E+00	1.02E−04	1.00E−02	1	2.64E+00

Table 9
Ranking results based on Holm-Bonferroni procedure for the compared algorithms on test functions from f_{41} to f_{47}.

Rank	Algorithm	z	p	θ	h	Score
1	SL-PSO	–	–	–	–	4.62E+00
2	MLCC	−1.07E+00	1.42E−01	5.00E−02	0	4.00E+00
3	CCPSO2	−1.32E+00	9.35E−02	2.50E−02	0	3.86E+00
4	DMS-L-PSO	−1.48E+00	6.88E−02	1.67E−02	0	3.76E+00
5	sep-CMA-ES	−2.80E+00	2.52E−03	1.25E−02	1	3.00E+00
6	EPUS-PSO	−4.95E+00	3.74E−07	1.00E−02	1	1.76E+00

statistically significant difference between the performances of DMS-PSO and SL-PSO on f_{13} to f_{40}, which basically confirms the results based on t-test. The ranking results for the high-dimensional problems, f_{41} to f_{47}, are presented in Table 9. From this table, we can conclude that SL-PSO has shown comparable performance to MLCC, CCPSO2 and DMS-PSO, the three algorithms designed for solving large-scale optimization problems, and outperforms sep-CMA-ES and EPUS-PSO. These results suggest that SL-PSO, despite its algorithmic simplicity and high computational efficiency, is promising also for solving large-scale optimization problems.

4. Conclusion

This paper has introduced a social learning PSO (SL-PSO) inspired by learning mechanisms in social learning of animals. Extensive experiments have been performed to compare the performance of the proposed SL-PSO with five representative PSO variants on 40 low-dimensional test functions and another five state-of-the-art algorithms for large-scale optimization on seven high-dimensional test functions. From these results, the following conclusions can be drawn. First, SL-PSO has been shown to work consistently well without fine tuning of the parameters on a large number of test problems with the search dimensionality varying from 30 to 1000. Moreover, SL-PSO has a higher computational efficiency in comparison with a few representative PSO variants. Compared to most modern meta-heuristics for optimization, SL-PSO is easy to implement, computationally efficient and requires no cumbersome fine-tuning of the control parameters. These properties make it very appealing for solving real-world problems, where little problem-specific knowledge is available and fine-tuning of the control parameters is less likely, if not impossible. In the future, we would like to examine performance of SL-PSO on real-world applications and extend the SL-PSO for multi-objective[16] or even many-objective optimization [37].

Acknowledgements

This work was supported in part by the Honda Research Institute Europe and the Joint Research Fund for Overseas Chinese, Hong Kong and Macao Scholars of the National Natural Science Foundation of China (Grant No. 61428302).

References

[1] A. Auger, N. Hansen, A restart CMA evolution strategy with increasing population size, Proceedings of IEEE Congress on Evolutionary Computation, vol. 2, IEEE, 2005, pp. 1769–1776.
[2] S. Baskar, P.N. Suganthan, A novel concurrent particle swarm optimization, Proceedings of IEEE Congress on Evolutionary Computation, vol. 1, IEEE, 2004, pp. 792–796.
[3] C.F. Bond, L.J. Titus, Social facilitation: a meta-analysis of 241 studies, Psychol. Bull. 94 (2) (1983) 265.
[4] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (6) (2006) 646–657.
[5] R. Brits, A.P. Engelbrecht, F. van den Bergh, Locating multiple optima using particle swarm optimization, Appl. Math. Comput. 189 (2) (2007) 1859–1883.
[6] L. Cagnina, M. Errecalde, D. Ingaramo, P. Rosso, An efficient particle swarm optimization approach to cluster short texts, Inform. Sci. 265 (2014) 36–49.
[7] F. Caraffini, F. Neri, L. Picinali, An analysis on separability for memetic computing automatic design, Inform. Sci. 265 (2014) 1–22.
[8] W.-N. Chen, J. Zhang, Y. Lin, N. Chen, Z.-H. Zhan, H. Chung, Y. Li, Y. Shi, Particle swarm optimization with an aging leader and challengers, IEEE Trans. Evolut. Comput. 17 (2) (2013) 241–258.
[9] R. Cheng, Y. Jin, A competitive swarm optimizer for large scale optimization, IEEE Trans. Cybernet (2014).
[10] R. Cheng, C. Sun, Y. Jin, A multi-swarm evolutionary framework based on a feedback mechanism, in: IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 718–724.
[11] R. Cheng, M. Yao, Particle swarm optimizer with time-varying parameters based on a novel operator, Appl. Math. Inform. Sci. 5 (2) (2011) 33–38.
[12] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evolut. Comput. 6 (1) (2002) 58–73.
[13] M. Daneshyari, G.G. Yen, Cultural-based multiobjective particle swarm optimization, IEEE Trans. Cybernet. 41 (2) (2011) 553–567.
[14] K. Dautenhahn, C.L. Nehaniv, A. Alissandrakis, Learning by experience from others social learning and imitation in animals and robots, in: Adaptivity and Learning: An Interdisciplinary Debate, Springer Verlag, 2003, pp. 217–421.
[15] M.A.M. de Oca, T. Stutzle, K. Van den Enden, M. Dorigo, Incremental social learning in particle swarms, IEEE Trans. Syst. Man Cybernet. Part B: Cybernet. 41 (2) (2011) 368–384.
[16] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, Hoboken, NJ, 2001.
[17] J. Derrac, S. Garcia, 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.
[18] M. Dorigo, C. Blum, Ant colony optimization theory: a survey, Theor. Comput. Sci. 344 (2) (2005) 243–278.
[19] W. Du, B. Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, Inform. Sci. 178 (15) (2008) 3096–3109.

[20] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of International Symposium on Micro Machine and Human Science, IEEE, 1995, pp. 39–43.
[21] J.L. Fernandez-Martinez, E. Garcia-Gonzalo, Stochastic stability analysis of the linear continuous and discrete PSO models, IEEE Trans. Evolut. Comput. 15 (3) (2011) 405–423.
[22] J. Fisher, R.A. Hinde, The opening of milk bottles by birds, British Birds 42 (347) (1949) 57.
[23] S. García, A. Fernández, J. Luengo, F. Herrera, A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability, Soft Comput. 13 (10) (2009) 959–977.
[24] Y.-J. Gong, J. Zhang, H. Chung, W.-n. Chen, Z.-H. Zhan, Y. Li, Y. Shi, An efficient resource allocation scheme using particle swarm optimization, IEEE Trans. Evolut. Comput. 16 (6) (2012) 801–816.
[25] N. Hansen, S. Kern, Evaluating the CMA evolution strategy on multimodal test functions, in: Proceedings of PPSN VIII, Springer, 2004, pp. 282–291.
[26] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evol. Comput. 9 (2) (2001) 159–195.
[27] C. Heyes, E. Ray, C. Mitchell, T. Nokes, Stimulus enhancement: controls for social facilitation and local enhancement, Learn. Motivat. 31 (2) (2000) 83–98.
[28] C.M. Heyes, Social learning in animals: categories and mechanisms, Biol. Rev. 69 (2) (1994) 207–231.
[29] N. Higashi, H. Iba, Particle swarm optimization with gaussian mutation, in: Proceedings of IEEE Swarm Intelligence Symposium, IEEE, 2003, pp. 72–79.
[30] S.-Y. Ho, H.-S. Lin, W.-H. Liauh, S.-J. Ho, OPSO: orthogonal particle swarm optimization and its application to task assignment problems, IEEE Trans. Syst. Man Cybernet. Part A: Syst. Humans 38 (2) (2008) 288–298.
[31] C.A. Hoare, Quicksort, Comput. J. 5 (1) (1962) 10–16.
[32] S.-T. Hsieh, T.-Y. Sun, C.-C. Liu, S.-J. Tsai, Solving large scale global optimization using improved particle swarm optimizer, in: Proceedings of IEEE Congress on Evolutionary Computation, IEEE, 2008, pp. 1777–1784.
[33] M. Hu, T. Wu, J.D. Weir, An adaptive particle swarm optimization with multiple adaptive methods, IEEE Trans. Evolut. Comput. 17 (5) (2013) 705–720.
[34] A. Husseinzadeh Kashan, M. Husseinzadeh Kashan, S. Karimiyan, A particle swarm optimizer for grouping problems, Inform. Sci. 252 (2013) 81–95.
[35] G. Iacca, F. Caraffini, F. Neri, E. Mininno, Single particle algorithms for continuous optimization, in: IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 1610–1617.
[36] Z. Ishaque, K. Salam, A deterministic particle swarm optimization maximum power point tracker for photovoltaic system under partial shading condition, IEEE Trans. Indust. Electron. 60 (2013) 3195–3206.
[37] H. Ishibuchi, N. Tsukamoto, Y. Nojima, Evolutionary many-objective optimization: a short review, in: Proceedings of IEEE Congress on Evolutionary Computation, IEEE, 2008, pp. 2419–2426.
[38] Y. Jin, B. Sendhoff, Fitness approximation in evolutionary computation – a survey, in: Proceedings of Genetic and Evolutionary Computation Conference, 2002, pp. 1105–1112.
[39] C.-F. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, IEEE Trans. Syst. Man Cybernet. Part B: Cybernet. 34 (2) (2004) 997–1006.
[40] D. Karaboga, B. Akay, A survey: algorithms simulating bee swarm intelligence, Artif. Intell. Rev. 31 (1–4) (2009) 61–85.
[41] J. Kennedy, Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance, Proceedings of IEEE Congress on Evolutionary Computation, vol. 3, IEEE, 1999, pp. 1931–1938.
[42] J. Kennedy, Bare bones particle swarms, in: Proceedings of IEEE Swarm Intelligence Symposium, IEEE, 2003, pp. 80–87.
[43] J. Kennedy, R. Eberhart, Particle swarm optimization, Proceedings of IEEE International Conference on Neural Networks, vol. 4, IEEE, 1995, pp. 1942–1948.
[44] J. Kennedy, R. Mendes, Population structure and particle swarm performance, Proceedings of IEEE Congress on Evolutionary Computation, vol. 2, IEEE, 2002, pp. 1671–1676.
[45] L. Lefebvre, The opening of milk bottles by birds: evidence for accelerating learning rates, but against the wave-of-advance model of cultural transmission, Behav. Process. 34 (1) (1995) 43–53.
[46] X. Li, Y. Yao, Cooperatively coevolving particle swarms for large scale optimization, IEEE Trans. Evol. Comput. 16 (2) (2011) 1–15.
[47] J. Liang, B. Qu, P. Suganthan, A.G. Hernández-Díaz, Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, 2012, Technical Report 201212.
[48] J. Liang, P. Suganthan, Dynamic multi-swarm particle swarm optimizer, in: Proceedings of IEEE Swarm Intelligence Symposium, IEEE, 2005, pp. 124–129.
[49] J.J. Liang, A. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evolut. Comput. 10 (3) (2006) 281–295.
[50] D. Lim, Y. Jin, Y.-S. Ong, B. Sendhoff, Generalizing surrogate-assisted evolutionary computation, IEEE Trans. Evolut. Comput. 14 (3) (2010) 329–355.
[51] B. Liu, L. Wang, Y.-H. Jin, F. Tang, D.-X. Huang, Improved particle swarm optimization combined with chaos, Solit. Fract. 25 (5) (2005) 1261–1271.
[52] F.-Q. Lu, M. Huang, W.-K. Ching, T.K. Siu, Credit portfolio management using two-level particle swarm optimization, Inform. Sci. 237 (2013) 162–175.
[53] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Trans. Evolut. Comput. 8 (3) (2004) 204–210.
[54] S. Mineka, M. Cook, Social learning and the acquisition of snake fear in monkeys, Social learning: Physhological and biological perspectives (1998) 51–73.
[55] R.W. Mitchell, A comparative-developmental approach to understanding imitation, in: Perspectives in Ethology, Springer, 1987, pp. 183–215.
[56] I. Montalvo, J. Izquierdo, R. Pérez, P.L. Iglesias, A diversity-enriched variant of discrete PSO applied to the design of water distribution networks, Eng. Optim. 40 (7) (2008) 655–668.
[57] M.A. Montes de Oca, T. Stützle, Towards incremental social learning in optimization and multiagent systems, in: Proceedings of the Conference Companion on Genetic and Evolutionary Computation, ACM, 2008, pp. 1939–1944.
[58] F. Neri, E. Mininno, G. Iacca, Compact particle swarm optimization, Inform. Sci. 239 (2013) 96–121.
[59] M. Neshat, G. Sepidnam, M. Sargolzaei, A.N. Toosi, Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications, Artif. Intell. Rev. (2012) 1–33.
[60] L. Palafox, N. Noman, H. Iba, Reverse engineering of gene regulatory networks using dissipative particle swarm optimization, IEEE Trans. Evolut. Comput. 17 (2013) 577–587.
[61] Y.V. Pehlivanoglu, A new particle swarm optimization method enhanced with a periodic mutation strategy and neural networks, IEEE Trans. Evolut. Comput. 17 (3) (2013) 436–452.
[62] M. Potter, K. De Jong, A cooperative coevolutionary approach to function optimization, Parallel Problem Solving from Nature-PPSN III (1994) 249–257.
[63] B. Qu, P. Suganthan, S. Das, A distance-based locally informed particle swarm model for multi modal optimization, IEEE Trans. Evolut. Comput. 17 (3) (2013) 387–402.
[64] A. Ratnaweera, S.K. Halgamuge, H.C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Trans. Evolut. Comput. 8 (3) (2004) 240–255.
[65] J. Robinson, S. Sinton, Y. Rahmat-Samii, Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna, Proceedings of IEEE Antennas and Propagation Society International Symposium, vol. 1, IEEE, 2002, pp. 314–317.
[66] R. Ros, N. Hansen, A simple modification in CMA-ES achieving linear time and space complexity, Parallel Problem Solving from Nature-PPSN X (2008) 296–305.
[67] R. Ruiz-Cruz, E. Sanchez, F. Ornelas-Tellez, A.G. Loukianov, R. Harley, Particle swarm optimization for discrete-time inverse optimal control of a doubly fed induction generator, IEEE Trans. Cybernet. 43 (2013) 1698–1709.

[68] S. Sastry, Nonlinear Systems: Analysis, Stability, and Control, vol. 10, Springer, New York, 1999.
[69] M. Setayesh, M. Zhang, M. Johnston, A novel particle swarm optimisation approach to detecting continuous, thin and smooth edges in noisy images, Inform. Sci. 246 (2013) 28–51.
[70] P. Shelokar, P. Siarry, V.K. Jayaraman, B.D. Kulkarni, Particle swarm and ant colony algorithms hybridized for improved continuous optimization, Appl. Math. Comput. 188 (1) (2007) 129–142.
[71] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: Proceedings of IEEE International Conference on Evolutionary Computation, IEEE, 1998, pp. 69–73.
[72] Y. Shi, R. Eberhart, Parameter selection in particle swarm optimization, in: Evolutionary Programming VII, Springer, 1998, pp. 591–600.
[73] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, IEEE, 1999, pp. 1945–1950.
[74] Y. Shi, R.C. Eberhart, Fuzzy adaptive particle swarm optimization, Proceedings of IEEE Congress on Evolutionary Computation, vol. 1, IEEE, 2001, pp. 101–106.
[75] Y. Shi et al, Particle swarm optimization: developments, applications and resources, Proceedings of IEEE Congress on Evolutionary Computation, vol. 1, IEEE, 2001, pp. 81–86.
[76] P.N. Suganthan, Particle swarm optimiser with neighbourhood operator, Proceedings of IEEE Congress on Evolutionary Computation, vol. 3, IEEE, 1999, pp. 1958–1962.
[77] C. Sun, J. Zeng, J.-S. Pan, S. Xue, Y. Jin, A new fitness estimation strategy for particle swarm optimization, Inform. Sci. 221 (2013) 355–370.
[78] K. Tang, X. Yao, P.N. Suganthan, C. MacNish, Y.-P. Chen, C.-M. Chen, Z. Yang, Benchmark functions for the CEC 2008 special session and competition on large scale global optimization, Nature Inspired Computation and Applications Laboratory, USTC, China, 2008.
[79] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Inform. Process. Lett. 85 (6) (2003) 317–325.
[80] 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, Inform. Sci. 270 (2014) 143–153.
[81] F. Van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evolut. Comput. 8 (3) (2004) 225–239.
[82] J. Vesterstrom, R. Thomsen, A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, Proceedings of IEEE Congress on Evolutionary Computation, vol. 2, IEEE, 2004, pp. 1980–1987.
[83] H. Wang, H. Sun, C. Li, S. Rahnamayan, J.-s. Pan, Diversity enhanced particle swarm optimization with neighborhood search, Inform. Sci. 223 (2013) 119–135.
[84] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, Q. Tian, Self-adaptive learning based particle swarm optimization, Inform. Sci. 181 (20) (2011) 4515–4538.
[85] A. Whiten, Primate culture and social learning, Cognit. Sci. 24 (3) (2000) 477–508.
[86] W. Xu, Z. Geng, Q. Zhu, X. Gu, A piecewise linear chaotic map and sequential quadratic programming based robust hybrid particle swarm optimization, Inform. Sci. 218 (2013) 85–102.
[87] Y. Yang, J.O. Pedersen, A comparative study on feature selection in text categorization, in: Proceedings of International Conference on Machine Learning, Morgan Kaufmann Publishers, 1997, pp. 412–420.
[88] Z. Yang, K. Tang, X. Yao, Differential evolution for high-dimensional function optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, IEEE, 2007, pp. 3523–3530.
[89] Z. Yang, K. Tang, X. Yao, Large scale evolutionary optimization using cooperative coevolution, Inform. Sci. 178 (15) (2008) 2985–2999.
[90] Z. Yang, K. Tang, X. Yao, Multilevel cooperative coevolution for large scale optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, IEEE, 2008, pp. 1663–1670.
[91] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evolut. Comput. 3 (2) (1999) 82–102.
[92] G.G. Yen, M. Daneshyari, Diversity-based information exchange among multiple swarms in particle swarm optimization, Int. J. Comput. Intell. Appl. 7 (01) (2008) 57–75.
[93] H.P. Young, Innovation diffusion in heterogeneous populations: contagion, social influence, and social learning, Am. Econ. Rev. 99 (5) (2009) 1899–1924.
[94] T.R. Zentall, Imitation in animals: evidence, function, and mechanisms, Cybernet. Syst. 32 (1–2) (2001) 53–96.
[95] Z.-H. Zhan, J. Zhang, Y. Li, H.-H. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst. Man Cybernet. Part B: Cybernet. 39 (6) (2009) 1362–1381.
[96] Z.-H. Zhan, J. Zhang, Y. Li, Y. Shi, Orthogonal learning particle swarm optimization, IEEE Trans. Evolut. Comput. 15 (6) (2011) 832–847.
[97] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evolut. Comput. 13 (5) (2009) 945–958.
[98] S.-Z. Zhao, J.J. Liang, P.N. Suganthan, M.F. Tasgetiren, Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, IEEE, 2008, pp. 3845–3852.
[99] J. Zhou, Z. Ji, L. Shen, Simplified intelligence single particle optimization based neural network for digit recognition, in: Chinese Conference on Pattern Recognition, IEEE, 2008, pp. 1–5.
[100] Z. Zhu, J. Zhou, Z. Ji, Y. Shi, DNA sequence compression using adaptive particle swarm optimization-based memetic algorithm, IEEE Trans. Evolut. Comput. 15 (5) (2011) 643–658.