



Multiple-strategy learning particle swarm optimization for large-scale optimization problems

Hao Wang¹ · Mengnan Liang¹ · Chaoli Sun¹ · Guochen Zhang¹ · Liping Xie¹

Received: 15 March 2020 / Accepted: 26 April 2020 / Published online: 26 May 2020
© The Author(s) 2020

Abstract

The balance between the exploration and the exploitation plays a significant role in the meta-heuristic algorithms, especially when they are used to solve large-scale optimization problems. In this paper, we propose a multiple-strategy learning particle swarm optimization algorithm, called MSL-PSO, to solve problems with large-scale variables, in which different learning strategies are utilized in different stages. At the first stage, each individual tries to probe some positions by learning from the demonstrators who have better performance on the fitness value and the mean position of the population. All the best probed positions, each of which has the best fitness among all positions probed by its corresponding individual, will compose a new temporary population. The temporary population will be sorted on the fitness values in a descending order, and will be used for each individual to find its demonstrators, which is based on the rank of the best probed solution in the temporary population and the rank of the individual in the current population, to learn using a new strategy in the second stage. The first stage is used to improve the exploration capability, and the second one is expected to balance the convergence and diversity of the population. To verify the effectiveness of MSL-PSO for solving large-scale optimization problems, some empirical experiments are conducted, which include CEC2008 problems with 100, 500, and 1000 dimensions, and CEC2010 problems with 1000 dimensions. Experimental results show that our proposed MSL-PSO is competitive or has a better performance compared with ten state-of-the-art algorithms.

Keywords Large-scale optimization · Multiple-strategy learning particle swarm optimization · Two-stage searching mechanism

Introduction

Optimization problems can be seen everywhere, for example, the optimization of the water distribution network [36], the

optimization strategy of the resource allocation [16], the task assignment [19], and many others [1, 12, 21, 24, 39–41, 48, 50, 72]. As a maximum optimization problem can be transferred to a minimum one by multiplying -1 , in this paper, only the minimization problems are considered. The mathematical model of a minimization problem can be described in the following:

$$\min f(\mathbf{x}) \quad (1)$$

$$\text{s.t. } \mathbf{x} \in \mathbb{R}^D, \quad (2)$$

where D is the dimension of the problem, and $f(\mathbf{x})$ is the objective of the optimization. Meta-heuristic optimization algorithms, including evolutionary algorithms [11, 14], and swarm optimization algorithms [15, 20, 26, 27, 31, 32, 44, 46, 56] have been paid more and more attention and applied successfully in many optimization problems [23, 27, 43, 52, 70] because of its ease of usage and independence on the characteristics of the optimization problems.

Hao Wang and Mengnan Liang are co-first authors.

✉ Chaoli Sun
chaoli.sun.cn@gmail.com

Hao Wang
h_wang_cn@163.com

Mengnan Liang
liangmn5@163.com

Guochen Zhang
imzgc@hotmail.com

Liping Xie
xieliping@tyust.edu.cn

¹ Department of Computer Science and Technology, Taiyuan University of Science and Technology, Taiyuan 030024, China

Traditional meta-heuristics have shown excellent abilities for solving lower dimensional optimization problems. However, their performances would deteriorate dramatically when the number of dimensions exceeds 100 [6], called large-scale optimization problems, because of the curse of dimensionality [34,59]. The search process will then stagnate into a local optimum and, therefore, result in a premature convergence. Thus, it is critical to enhance the diversity of the population so as to improve the exploration capability for tackling the large-scale optimization problems. Some approaches have been proposed for solving problems with large-scale dimensions, which can be divided into two categories: cooperative coevolution (CC) methods and new learning strategies.

1. The cooperative coevolutionary methods adopt the divide-and-conquer strategy, which decomposes the population into a number of subpopulations and solves them independently. Many algorithms based on cooperative coevolutionary frameworks for large-scale optimization problems have been proposed, such as cooperative coevolutionary genetic algorithms [2,3,22,42,55,69] cooperative coevolutionary PSO (CCPSO) [30], and cooperative coevolutionary DE (DECC) [29,37,51,64–67], etc. The methods are efficient when the optimization problems are separable. However, the performance is not so good when the problems are nonseparable. Therefore, different decomposition strategies, such as random dynamic grouping [66], multilevel dynamic grouping [67], and differential grouping [29,38], have been proposed. Also, it is obvious that the performance of CC algorithms is highly sensitive to the decomposition strategies for different classes of optimization problems.
2. In each meta-heuristic algorithm, the learning strategy plays a significant role to find an optimal solution. Therefore, researchers try to develop different learning strategies for meta-heuristic algorithms so as to enhance the exploration capability of the population, which aims to promote the chances of escaping from local optima [5–8,17,18,25,28,47,58]. The performance of exploration is improved by keeping the diversity of the population. However, more evaluations on the objective functions are required for the final convergence.

In recent years, the surrogate-assisted evolutionary algorithms, such as SA-COSO [54], SHPSO [68], and MGP-SLPSO [57], have been paid attention for solving computationally expensive high-dimensional problems (generally not more than 100 dimensions). However, not so many methods have been proposed for solving computationally expensive large-scale optimization problems. Ivanoe De Falco et al. [9,10] decomposed the large-scale optimization problems into some lower-dimensional sub-problems and optimized

using the surrogate-assisted optimization algorithms and parallel computing techniques. Sun et al. [53] proposed to solve large-scale optimization problems by a modified PSO algorithm assisted by the fitness estimation strategy. Generally, the number of exact fitness evaluation in the surrogate-assisted evolutionary algorithms for solving a computationally expensive problem is very limited, which will not be considered in this paper. That is, only the large-scale optimization problems with cheap fitness evaluation are considered to be solved in this paper, in which a large number of fitness evaluations are allowed.

To balance between the exploration and exploitation in solving large-scale optimization problems, in this paper, we propose a multiple strategy learning particle swarm optimization (MSL-PSO) method, in which two steps with different learning strategies are proposed to generate a new population. At the first step, the current population will be sorted from the worst to the best based on the fitness values, and each individual will probe different positions by learning from its demonstrators and the mean position of the current population. The best position probed by each individual will be kept and compose a temporary population, which will also be sorted from the worst to the best based on their fitness values. Each individual will update its position by learning demonstrators from different sub-sets of the temporary population, which is used to balance the convergence and diversity. Similar to those learning strategies that have been proposed, such as SL-PSO [6], CSO [5], and DEGLSO [63], etc., our MSL-PSO method also focuses on the learning strategy that concerns the trade-off between the convergence and diversity of the population.

The remainder of this paper is organized as follows. The section “Related work” gives a brief introduction on the canonical particle swarm optimization (PSO) algorithm and the methods for solving the large-scale optimization problems. Our proposed MSL-PSO algorithm is described in detail in the section “The proposed algorithm”. In the section “Experimental studies”, experiments are conducted to verify the effectiveness and efficiency of MSL-PSO by comparison with ten state-of-the-art algorithms. Finally, the conclusion of this paper is given in the section “Conclusion”.

Related work

Particle swarm optimization

The particle swarm optimization (PSO), which simulates the bird flocking or fish schooling, was proposed by Kennedy and Eberhart [13], is one of the swarm optimization algorithms. In PSO, each individual has its own velocity and position, which will be updated using the following equations:

$$v_{id}(t+1) = \omega v_{id}(t) + c_1 r_1 (p_{id}(t) - x_{id}(t)) + c_2 r_2 (g_d(t) - x_{id}(t)) \quad (3)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1), \quad (4)$$

where $\mathbf{v}_i(t) = (v_{i1}(t), v_{i2}(t), \dots, v_{iD}(t))$ and $\mathbf{x}_i(t) = (x_{i1}(t), x_{i2}(t), \dots, x_{iD}(t))$ represent the velocity and position of individual i at t th generation, respectively. $\mathbf{p}_i(t) = (p_{i1}(t), p_{i2}(t), \dots, p_{iD}(t))$ and $\mathbf{g}(t) = (g_1(t), g_2(t), \dots, g_D(t))$ are the best historical position of individual i and the swarm, respectively. w is called the inertia weight, c_1 and c_2 are two cognitive parameters, and r_1 and r_2 are two random number generated uniformly in the range of $[0, 1]$. The PSO algorithm has been shown better convergence performance; however, it is not good for solving large-scale optimization problems [6]. Therefore, different PSO variants, such as CSO [5], SL-PSO [6], LLSO [61] and DEGLSO [63], have been proposed, in which different learning strategies were utilized in the swarm optimization algorithms to improve the diversity of PSO so as to enhance the exploration capability.

Optimization of the large-scale problems

Generality, the meta-heuristic algorithms proposed for solving large-scale optimization problems can be classified into two categories. One is the cooperative coevolutionary meta-heuristic algorithms, and the other is the meta-heuristic methods with efficient learning strategies.

Cooperative coevolutionary algorithms

The cooperative coevolution (CC) mechanism divides the large-scale problem into a number of small subproblems, and then optimize them separately using the canonical meta-heuristics. Generally, there are also two categories of CC evolutionary algorithms according to the decomposition strategies: the static and dynamic decomposition strategies. The static decomposition strategy first detects the relationship between variables and then decompose them in a fixed way before the optimization. For example, Omidvar et al. [37] proposed a differential grouping (DG) strategy, in which the problem is separated into a number of small subproblems. Mei et al. [33] extended the DG method to identify the independent subproblems to be optimized by a covariance matrix adaptation evolution strategy. Different to the static decomposition strategy, the variables are decomposed to different subproblems at different generations, which can be further classified into random-based decomposition and learning-based decomposition. Li et al. [29] proposed CCPSO2 for large-scale optimization problems, in which the random grouping was adopted to decompose the variables into subcomponents dynamically. However, the performance will be deteriorated when the problem is nonseparable. Ray

and Yao [45] proposed an adaptive variable partitioning based on the correlation, which was utilized in the cooperative coevolutionary algorithms to deal with nonseparable problems.

The decomposition strategy plays significant importance in the cooperative coevolutionary algorithms. Poor decomposition will deteriorate the performance the algorithms, and also, it will be inefficient when all variances of the problem are correlated to each other.

New learning strategies for meta-heuristic algorithms

Different to the cooperative coevolutionary algorithms, some learning strategies are proposed to be put into the meta-heuristic algorithms to improve the diversity of the population, and thus enhance the exploration capability of the algorithms. In our method, we focus on studying a new learning strategy for PSO to find a global optimal solution for the large-scale optimization problems. Therefore, only PSO variants for large-scale optimization were reviewed in this section. Cheng and Jin [5] proposed a competitive swarm optimizer (CSO) algorithm, in which a random competition strategy was utilized and any individual with a better fitness value in a pair will be the winner. Instead of learning from the personal best position of the individual and of the swarm, in CSO, the loser in the pair will learn from its winner. Inspired by CSO, Yang et al. [62] proposed a segment-based predominant learning swarm optimizer (SPLSO), in which the dimension will be divided into a number of segments and variables in different segments will be evolved by learning from different exemplars. The social learning PSO [6] was also proposed by Cheng and Jin, in which the population will be sorted according to the fitness and each individual will learn from its demonstrators [4] who have better fitness values than this individual. Later, Yang et al. [61] separated the population into a number of levels based on the fitness values, and each individual will learn from particles in two higher layers.

The proposed algorithm

The trade-off between the convergence and diversity of the population is crucial for meta-heuristic algorithms to solve large-scale optimization problems, because the search space will increase exponentially when the dimension of the problem increases [62], which will put significant challenges to find the global optimal solution. PSO has been shown to be implemented easily and has quick convergence capability. However, the diversity of the population will be lost quickly after some generations because of the quick convergence of the algorithm. Therefore, the PSO algorithm is not efficient to solve large-scale optimization problems. Some PSO variants,

such as CSO [5], SL-PSO [6], SPLSO [62], and LLSO [61], have put forward different learning strategies to improve the diversity of the population to solve large-scale optimization problems. In this paper, we propose a multi-strategy learning particle swarm optimization (MSL-PSO) method, in which the idea of social learning proposed in [6] is adopted to update the position of each individual in the population. However, different to the SL-PSO algorithm, two stages with different learning strategies are used to generate a new population. In the first stage, each individual will probe some positions by learning from its demonstrators who have better fitness values than this individual and the mean position of the population. The best probed position, which has the minimal fitness value among all of its probed positions, will be kept. All the best probed positions at current generation will compose a temporary population, which will be sorted from worst to best on the fitness values. Then, in the second stage, a new strategy, which is used to balance the diversity and convergence, will be used to update the velocity and position of each individual. In the following, we will give a detailed description of the proposed method.

The overall framework of MSL-PSO

Algorithm 1 gives a pseudocode of the MSL-PSO algorithm. A population Pop will be initialized, and the fitness of each individual in Pop will be evaluated. While the stopping criteria is not met, the following process will be repeated: all individuals will be sorted on the fitness values in a descending order. Each individual will probe K_{max} positions by learning from its demonstrators and the mean position of the current population. The best probed position, which has the minimum fitness value among K_{max} positions, will be kept for each individual (lines 6–7). All of these best probed positions will compose a temporary population NPop. Then, individuals in the NPop will also be sorted on their fitness values in a descending order. Afterwards, each individual i will find two sub-sets of NPop, one is composed of all individuals whose rank in NPop is between the rank of individual i in the sorted NPop and that of individual i in the sorted Pop, and the other is composed of all individuals whose rank is larger than that of individual i in the sorted NPop, for selecting demonstrators to update its velocity using the strategy proposed in our method (line 11). At each generation, an individual will explore $K_{max} + 1$ different positions by the two learning criteria, and all of these $K_{max} + 1$ positions will be evaluated using the objective function. Therefore, there will be $K_{max} + 1$ fitness evaluations for each solution at each generation, and the total number of fitness evaluations will be $(K_{max} + 1) * NP$, where NP is the size of current population.

Algorithm 1: The pseudocode of MSL-PSO

Input: Population size NP , the number of maximal fitness evaluations MAX_FES ;
Output: The optimal solution and its fitness;
1 Initialization of a population Pop;
2 Evaluate the fitness of each individual in Pop, $fes = NP$;
3 **while** $fes \leq MAX_FES$ **do**
4 Sort the population in a descending order;
5 **for** $i = 1 : NP$ **do**
6 Probe K_{max} positions using the social learning technique proposed in [6] for individual i ; (**Refer to Algorithm 2**)
7 Evaluate these K_{max} positions and keep the best solution among these K_{max} solutions;
8 **end for**
9 Sort the new population NPop, which is composed of the best probed position of each individual in Pop, in a descending order;
10 **for** $i = 1 : NP$ **do**
11 Find two sub-sets in the new population NPop for the social learning of individual i , and the population Pop will be updated; (**Refer to Algorithm 3**)
12 **end for**
13 $fes = fes + (K_{max} + 1) * NP$;
14 **end while**
15 Output the optimal solution and its fitness value;

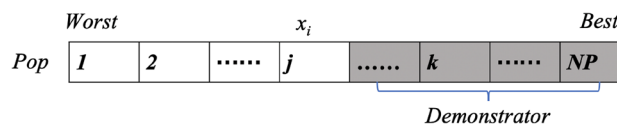


Fig. 1 Population sorting

Position probing

Algorithm 2 gives the detail on how an individual probes different positions by social learning technique proposed in [6]. Similar to SL-PSO [6], all individuals in the current population will be sorted in a descending order based on the fitness value, which is shown in Fig. 1. Suppose that the rank of individual i is j , and then, individuals that have larger rank than j are all demonstrators of individual i .

For each individual i , K_{max} positions will be probed using Eqs. (5) and (6) by learning from the demonstrators to improve the exploration capability of the population:

$$vv_{id}^k = r_1^k * v_{id}(t) + r_2^k * (x_{jd}(t) - x_{id}(t)) + \phi * r_3^k * (\bar{x}_d(t) - x_{id}(t)) \tag{5}$$

$$xx_{id}^k = x_{id}(t) + vv_{id}^k, \tag{6}$$

where $k = 1, 2, \dots, K_{max}$; K_{max} is the maximum number of probation for each individual. $\mathbf{vv}_i^k = (vv_{i1}^k, vv_{i2}^k, \dots, vv_{iD}^k)$ and $\mathbf{xx}_i^k = (xx_{i1}^k, xx_{i2}^k, \dots, xx_{iD}^k)$ represent the velocity and position that individual i learns from its demonstrators at k th time, respectively. $\mathbf{v}_i(t) = (v_{i1}, v_{i2}, \dots, v_{iD})$ and $\mathbf{x}_i(t) = (x_{i1}, x_{i2}, \dots, x_{iD})$ are the velocity and position of

individual i at generation t , respectively. $\bar{\mathbf{x}}(t) = \frac{1}{\text{NP}} \sum_{i=1}^{\text{NP}} \mathbf{x}_i$ is the mean position of the population at generation t . r_1^k, r_2^k , and r_3^k are random numbers uniformly generated in the range of $[0, 1]$ at k th time. ϕ is a constant called the social learning probability which is used to define the degree to learn from the mean position of the population. Note that the K_{\max} positions have little probability to be same, because the individual will choose different demonstrators to learn on each dimension. It can be easily imaged that more positions are probed by learning from demonstrators, more chances it will have to find a better solution at each generation. However, the computational resources will be quickly exhausted, because too many fitness evaluations will be consumed at each generation, which, in turn, will impede the population to explore the search space and, correspondingly, will not be able to find a good optimal solution. On the other hand, the less positions are probed, the less probability it has to find a better solution at each generation. The best probed position with the minimum fitness value among K_{\max} solutions will be kept and denoted as \mathbf{xc} (line 9 in Algorithm 2), and its corresponding velocity is denoted as \mathbf{vc} .

Algorithm 2: Position probing

Input: individual i ;
Output: the best probed position \mathbf{xc}_i ;

```

1 for  $k = 1 : K_{\max}$  do
2   for  $d=1:D$  do
3     Randomly select one individual from its
       demonstrators;
4     Update the velocity on  $d$ th dimension using Eq. (5);
5   end for
6   Generate  $k$ th candidate position using Eq. (6);
7 end for
8 Evaluate the fitness values of these candidate positions;
9  $\mathbf{xc}_i = \text{argmin}\{f(\mathbf{xx}_1), f(\mathbf{xx}_2), \dots, f(\mathbf{xx}_{K_{\max}})\}$ ;
```

Position updating

The convergence and diversity are two key factors in finding the global optimal solutions for large-scale optimization problems. A good performance on the diversity can improve the exploration capability, while a good performance on the convergence can speed up locating at the optimal position. Therefore, in the second stage of MSL-PSO, we propose a new strategy, in which the demonstrators of each individual are selected from two sub-sets of NPop, which is composed of all best probed positions, i.e., $\text{NPop} = \{\mathbf{xc}_1, \mathbf{xc}_2, \dots, \mathbf{xc}_{\text{NP}}\}$, to update the velocity of each individual in the population Pop. Equations (7) and (8) show how to update the velocity and position of each individual in the stage of position

updating for each individual:

$$v_{id}(t + 1) = r_1 * v_{cid} + r_2 * (xc_{jd} - xc_{id}(t)) + \phi * r_3 * (xc_{kd} - xc_{id}(t)) \tag{7}$$

$$x_{id}(t + 1) = xc_{id} + v_{id}(t + 1). \tag{8}$$

In Eqs. (7) and (8), $\mathbf{vc}_i = (vc_{i1}, vc_{i2}, \dots, vc_{iD})$ and $\mathbf{xc}_i = (xc_{i1}, xc_{i2}, \dots, xc_{iD})$ are the velocity and position, respectively, of the best probed position for individual i . j and k represent the two demonstrators in NPop for individual i to learn on dimension d . ϕ is the social learning probability which has the same meaning to that in Eq. (5).

Figure 2 gives a simple example to show how to select two demonstrators in the second stage. In Fig. 2, demonstrator_1 and demonstrator_2 represent two sub-sets for individual i to choose demonstrators to learn, respectively. For each individual i , we have its rank in Pop and rank in NPop. When the rank of the best probed position of individual i in the NPop is worse than the rank of the individual i in the Pop, it means that it falls behind more other individuals in the NPop compared to it does in the Pop. For example in Fig. 2a, the rank of \mathbf{xc}_i in NPop is k , and the rank of $\mathbf{x}_i(t)$ in Pop is j , $k < j$, then the demonstrators between k and j will be used to guide the individual to exploit a better solution. Conversely, when the rank of the best probed position of an individual i in NPop is better than that of the individual i in Pop, it means that the best probed position has a better performance among NPop than the individual i in Pop. To prevent premature convergence, we also learn from some losers of the best probe solution. Seen from Fig. 2b, we can see that $j < k$, and the losers between j and k will be selected as one of the demonstrators. The other demonstrator is selected from the sub-set demonstrator_2 which is composed of all individuals that have better fitness than the best probed position.

Algorithm 3 gives the pseudocode of the position updating. Each individual i will update its velocity and position according to Eqs. (7) and (8), respectively. The global best position $gbest$ will be updated if a new position is better than it.

Algorithm 3: Position updating

Input: the best probe position of individual i (\mathbf{xc}_i);
Output: the position of individual i at generation $t + 1$;

```

1 Update the velocity and position using Eqs. (7) and (8),
  respectively;
2 Evaluate the fitness value of individual  $i$ ;
3 if  $f(\mathbf{x}_i(t + 1)) < f(gbest)$  then
4   |  $gbest = \mathbf{x}_i(t + 1)$ ;
5   |  $f(gbest) = f(\mathbf{x}_i(t + 1))$ ;
6 end if
```

Table 1 Summary of the CEC2008 benchmark functions

	Function name	Properties	Search range
F1	Shifted sphere function	Unimodal; shifted; separable; scalable;	$[-100, 100]^D$
F2	Schwefel’s problem	Unimodal; shifted; non-separable; scalable;	$[-100, 100]^D$
F3	Shifted Rosenbrok’s function	Multi-modal; shifted; non-separable; scalable;	$[-100, 100]^D$
F4	Shifted Rastrigin’s function	Multi-modal; shifted; separable; scalable;	$[-5, 5]^D$
F5	Shifted Griewank’s function	Multi-modal; shifted; non-separable; scalable;	$[-600, 600]^D$
F6	Shifted Ackley’s function	Multi-modal; shifted; separable; scalable;	$[-32, 32]^D$
F7	Fast fractal ‘double dip’ function	Multi-modal; Non-separable; Scalable;	$[-1, 1]^D$

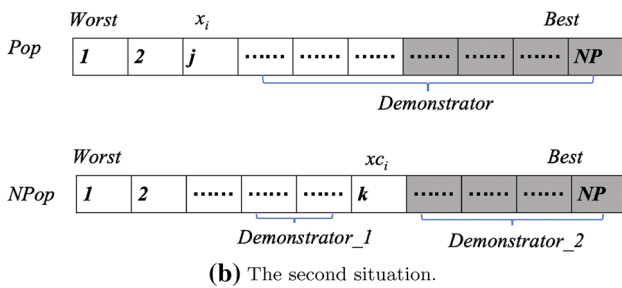
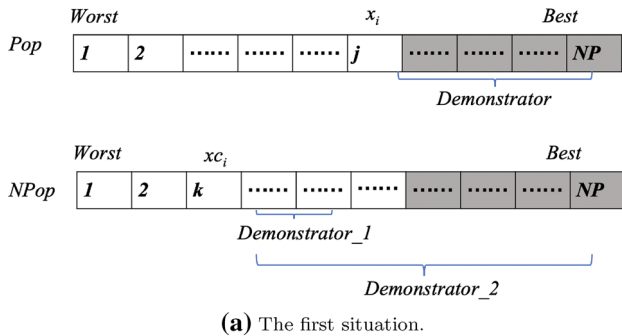


Fig. 2 A simple example to show the method in the second stage

Experimental studies

Experimental setup and benchmark functions

To verify the performance of MSL-PSO, a series of experiments are conducted on CEC2008 and CEC2010 large-scale benchmark problems. The main characteristics of these two function sets are summarized in Tables 1 and 2.

The parameter settings of MSL-PSO are given in the following: the population size NP is set to 100 if the dimension is not larger than 100, and otherwise, $NP = M + \lfloor (D/10) \rfloor$, where $M = 100$ and D is the dimension of the problem. The social learning probability ϕ is set to $D/M * 0.01$. The number of dimensionality D is set to 100, 500, and 1000 for CEC2008 test problems and 1000 for CEC2010 benchmark problems, respectively. The algorithm will be run 20 times independently on each problem. The stopping criteria are

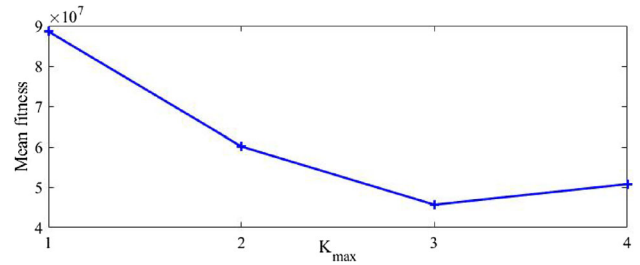


Fig. 3 The experimental results on the 1000-dimensional CEC2010 F9 problem obtained by the proposed method with different maximum number of candidate positions

that the maximum number of function evaluation reaches $3000 * D$. Generally, more candidate positions provided, more chances to find a better optimal solution. However, the number of fitness evaluation will be consumed much quicker if more candidate positions are considered. On the contrary, less positions are probed, less probability it has to find a better solution at each generation. To see which value is best for assisting the proposed method to obtain good optimal solutions, we compare the mean optimal solutions of the 1000-dimensional CEC2010 F9 problem obtained by the proposed method using different maximum number of candidate positions. Figure 3 gives the mean results of 20 independent runs with 1, 2, 3, and 4 positions probed by each individual at each generation, respectively. From Fig. 3, we can see that the performance of MSL-PSO is best when the maximum number of probed position is set to 3. Therefore, in our experiments, only three positions are probed for each individual by learning from its demonstrators, i.e., $K_{max} = 3$.

Comparisons to the MSL-PSO variants

From the detailed description of MSL-PSO, we can see that there are two stages to generate a new position for each individual. To see the performance of our proposed method, we first compare the results obtained on F6 and F9, which are multi-model and uni-model, respectively, with two MSL-PSO variants, in which one variant uses the first stage of MSL-PSO only (denoted as mSL-PSO) and the other one

Table 2 Summary of the CEC2010 benchmark functions

	Function name	Properties	Search range
F1	Shifted elliptic function	Unimodal; shifted; separable; scalable;	$[-100, 100]^D$
F2	Shifted Rastrigin's function	Multi-modal; shifted; separable; scalable;	$[-5, 5]^D$
F3	Shifted Ackley's function	Multi-modal; shifted; separable; scalable;	$[-32, 32]^D$
F4	Single-group shifted and m -rotated elliptic function	Unimodal; shifted; single-group m -rotated; single-group m -rotated;	$[-100, 100]^D$
F5	Single-group shifted and m -rotated Rastrigin's function	Multi-modal; shifted; single-group m -rotated; single-group m -non-separable;	$[-5, 5]^D$
F6	Single-group shifted and m -rotated Ackley's function	Multi-modal; shifted; single-group m -rotated; single-group m -non-separable;	$[-32, 32]^D$
F7	Single-group shifted and m -dimensional Schwefel's problem	Unimodal; shifted; single-group m -non-separable;	$[-100, 100]^D$
F8	Single-group shifted and m -dimensional Rosenbrock's function	Multi-modal; shifted; single-group m -non-separable;	$[-100, 100]^D$
F9	$\frac{D}{2m}$ -group shifted and m -rotated elliptic function	Unimodal; shifted; $\frac{D}{2m}$ -group m -rotated; $\frac{D}{2m}$ -group m -Non-separable	$[-100, 100]^D$
F10	$\frac{D}{2m}$ -group shifted and m -rotated Rastrigin's function	Multi-modal; shifted; $\frac{D}{2m}$ -group m -rotated; $\frac{D}{2m}$ -group m -Non-separable	$[-5, 5]^D$

replaces the strategy used in the second stage with the strategy proposed in SL-PSO (denoted as SL-PSOs). Table 3 gives the statistical results of these algorithms on CEC2010 F6 and F9 benchmark problems with 1000 dimensions. Each algorithm is run independently for 20 times on each test problem, and the Wilcoxon rank sum test with a significance level of 0.05 is applied to assess whether the performance of a solution obtained by one of the two compared algorithms is expected to be better than the other [60]. In Table 3, '+', '≈', and '-' represent that MSL-PSO is significantly better, equivalent

to, and worse than the compared algorithms, respectively, according to the Wilcoxon rank sum test on the mean fitness values. The best mean optimal solution on each benchmark problem is highlighted with an underline. From Table 3, we can see that our proposed MSL-PSO can obtain better results than mSL-PSO and SL-PSOs on both of these problems, which showed that the method using two stages is effective to solve large-scale optimization problems.

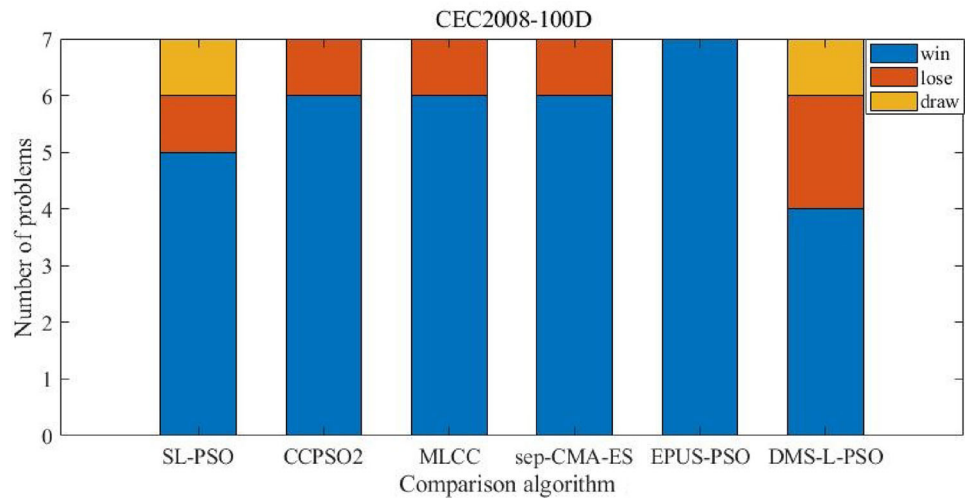
Table 2 continued

	Function name	Properties	Search range
F11	$\frac{D}{2m}$ -group Shifted and m -rotated Ackley's function	Multi-modal; Shifted; $\frac{D}{2m}$ -group m -rotated; $\frac{D}{2m}$ -group m -Non-separable	$[-32, 32]^D$
F12	$\frac{D}{2m}$ -group Shifted and m -dimensional Schwefel's problem	Unimodal; Shifted; $\frac{D}{2m}$ -group m -Non-separable	$[-100, 100]^D$
F13	$\frac{D}{2m}$ -group Shifted and m -dimensional Rosenbrock's function	Multi-modal; Shifted; $\frac{D}{2m}$ -group m -Non-separable	$[-100, 100]^D$
F14	$\frac{D}{m}$ -group shifted and m -rotated elliptic function	Unimodal; shifted; $\frac{D}{m}$ -group m -rotated ; $\frac{D}{m}$ -group m -Non-separable	$[-100, 100]^D$
F15	$\frac{D}{m}$ -group shifted and m -rotated Rastrigin's function	Multi-modal; shifted; $\frac{D}{m}$ -group m -rotated ; $\frac{D}{m}$ -group m -Non-separable	$[-5, 5]^D$
F16	$\frac{D}{m}$ -group shifted and m -rotated Ackley's function	Multi-modal; shifted; $\frac{D}{m}$ -group m -rotated ; $\frac{D}{m}$ -group m -Non-separable	$[-32, 32]^D$
F17	$\frac{D}{m}$ -group shifted m -dimensional Schwefel's function	Unimodal; shifted; $\frac{D}{m}$ -group m -Non-separable ;	$[-100, 100]^D$
F18	$\frac{D}{m}$ -group shifted m -rotated Rosenbrock's function	Multi-modal; shifted; $\frac{D}{m}$ -group m -Non-separable	$[-100, 100]^D$
F19	Shifted Schwefel's problem	Unimodal; shifted; fully-nonseparable	$[-100, 100]^D$
F20	Shifted Schwefel's problem	Multi-modal; shifted; fully-nonseparable	$[-100, 100]^D$

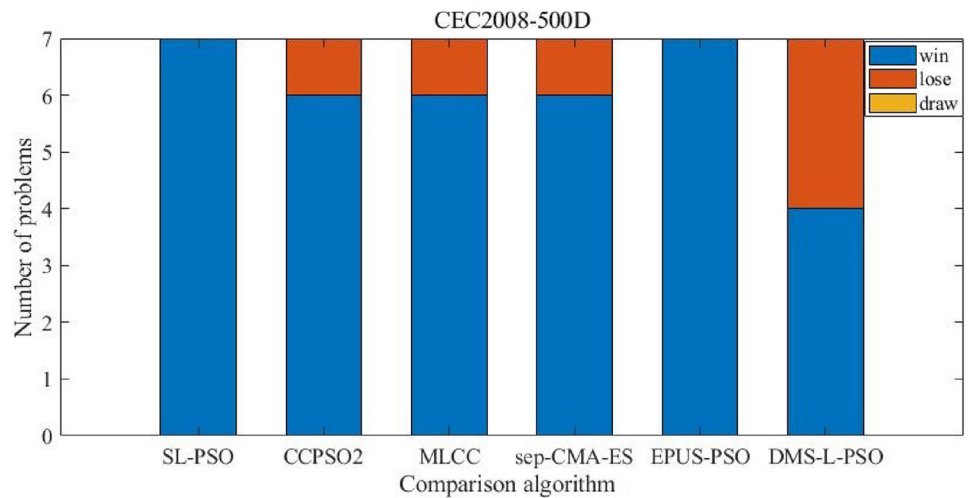
Table 3 Comparisons on the mean values and standard deviation between MSL-PSO and its variants on CEC2010 F6 and F9 test problems with 1000 dimensions ($D = 1000$)

		mSL-PSO	SL-PSOs	MSL-PSO
F6	Mean	1.42E-07 (+)	1.48E-07 (+)	9.07E-08
	Std	2.72E-09	7.14E-09	7.11E-01
F9	Mean	4.53E+07 (+)	6.96E+07 (+)	1.23E+07
	Std	2.40E+06	6.97E+06	1.13E+06

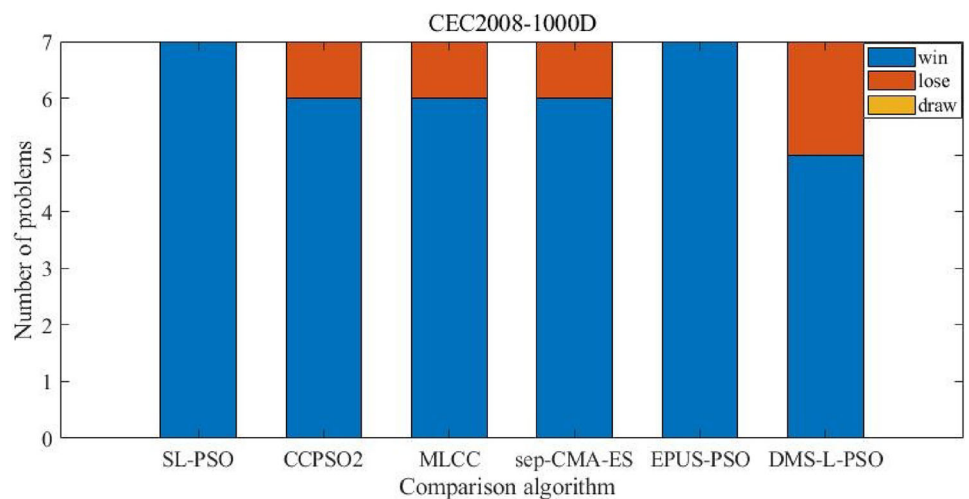
Fig. 4 Comparison on CEC 2008



(a) CEC 2008 100-dimensional problems



(b) CEC 2008 500-dimensional problems



(c) CEC 2008 1000-dimensional problems

Table 4 Comparisons of the mean value and standard deviation between MSL-PSO and other algorithms on CEC2008 test problems with 100 dimensions ($D = 100$)

		SL-PSO	CCPSO2	MLCC	Sep-CMA-ES	EPUS-PSO	DMS-L-PSO	MSL-PSO
F1	Mean	1.09e−27	7.73e−14	6.82e−14	9.02e−15	7.47e−01	0.00e−00	6.97e−28
	Std	3.50e−28	3.23e−14	2.32e−14	5.53e−15	1.70e−01	0.00e−00	3.14e−28
F2	Mean	9.45e−06	6.08e+00	2.53e+01	2.31e+01	1.86e+01	3.65e+00	5.03e−06
	Std	4.97e−06	7.83e+00	8.73e+00	1.39e+01	2.26e+00	7.30e−01	1.70e−06
F3	Mean	5.74e+02	4.23e+02	1.50e+02	4.31e+00	4.99e+03	2.83e+02	9.15e+01
	Std	1.67e+02	8.65e+02	5.72e+01	1.26e+01	5.35e+03	9.40e+02	1.85e+01
F4	Mean	7.46e+01	3.98e−02	4.39e−13	2.78e+02	4.71e+02	1.82e+02	1.18e+02
	Std	1.21e+01	1.99e−01	9.21e−14	3.43e+01	5.94e+01	2.16+01	9.34e+01
F5	Mean	0.00e+00	3.45e−03	3.41e−14	2.96e−04	3.72e−01	0.00e−00	0.00e+00
	Std	0.00e+00	3.45e−03	7.87e−14	1.48e−03	5.60e−02	0.00e−00	0.00e+00
F6	Mean	2.10e−14	1.44e−13	1.11e−13	2.12e+01	2.06e+00	0.00e−00	1.78e−14
	Std	5.22e−15	3.06e−14	7.87e−15	4.02e−01	4.40e−01	0.00e−00	7.72e−15
F7	Mean	−1.48e+03	−1.50e+03	−1.54e+03	−1.39e+03	−8.55e+02	−1.14e+03	−2.45e+03
	Std	1.90e+01	1.04e+01	2.52e+00	2.64e+01	8.48e+00	8.48e+00	1.00e+00

Table 5 Comparisons of the mean value and standard deviation between MSL-PSO and other algorithms on CEC2008 test problems with 500 dimensions ($D = 500$)

		SL-PSO	CCPSO2	MLCC	Sep-CMA-ES	EPUS-PSO	DMS-L-PSO	MSL-PSO
F1	Mean	7.24e−24	7.73e−14	4.30e−13	2.25e−14	8.45e+01	0.00e+00	6.83e−24
	Std	2.20e−25	3.23e−14	3.31e−14	6.10e−15	6.40e+00	0.00e+00	3.30e−25
F2	Mean	3.47e+01	5.69e+01	6.67e+01	2.12e+02	4.35e+01	6.89e+01	3.11e+01
	Std	1.03e+00	4.21e+01	5.70e+00	1.74e+01	9.51e−01	2.01e+00	7.14e−01
F3	Mean	6.10e+02	7.24e+02	9.25e+02	2.93e+02	5.77e+04	4.67e+07	4.85e+02
	Std	1.87e+02	1.54e+02	1.73e+02	3.59e+01	8.04e+03	5.87e+06	3.43e+02
F4	Mean	2.72e+03	3.98e−02	1.79e−11	2.18e+03	3.49e+03	1.61e+03	4.34e+03
	Std	3.25e+02	1.99e−01	6.31e−11	1.51e+02	1.11e+02	1.04e+02	5.22e+02
F5	Mean	3.33e−16	1.18e−03	2.13e−13	7.88e−04	1.64e+00	0.00e+00	8.76e−18
	Std	0.00e+00	4.61e−03	2.48e−14	2.82e−03	4.69e−02	0.00e+00	0.00e+00
F6	Mean	1.46e−13	5.34e−13	5.34e−13	2.15e+01	1.64e+00	2.00e+00	3.46e−14
	Std	2.95e−15	8.61e−14	7.01e−14	3.10e−01	4.49e−01	9.66e−02	5.01e−16
F7	Mean	−5.94e+03	−7.23e+03	−7.34e+03	−6.37e+03	−3.51e+03	−4.20e+03	−7.78e+03
	Std	1.72e+02	4.64e+01	8.03e+00	8.59e+00	1.29e+01	1.29e+01	7.11e+00

Comparisons to other state-of-the-art algorithms

In our experimental analysis, we further compare the results of MSL-PSO with ten state-of-the-art algorithms, which are shown in the following, to verify the performance of our proposed MSL-PSO algorithm. The best mean optimal solution on each benchmark problem is highlighted with an underline. Note that except the result of CSO, SL-PSO, and CCPSO2, all other results of the comparison algorithms are copied from their corresponding paper.

1. DECC-G [66]: Instead of using a static grouping, the optimization problem is randomly decomposed into k

subproblems in the decision space, and then co-evolute to find the global optimal solution.

2. MLCC [67]: The decomposer is selected by a self-adapted mechanism based on the historic performance at the start of each cycle.
3. DECC-DG [29]: A differential grouping is utilized to uncover the underlying interaction structure of the decision variables, and then, a number of subproblems are formed.
4. CSO [5]: A new competitive learning strategy for PSO was proposed to solve the problems with large-scale dimensions, in which every two individuals will be com-

Table 6 Comparisons of the mean value and standard deviation between MSL-PSO and other algorithms on CEC2008 test problems with 1000 dimensions ($D = 1000$)

		SL-PSO	CCPSO2	MLCC	Sep-CMA-ES	EPUS-PSO	DMS-L-PSO	MSL-PSO
F1	Mean	7.10e−23	5.18e−13	8.46e−13	7.81e−15	5.53e+02	0.00e+00	9.33e-25
	Std	1.40e−24	9.61e−14	5.01e−14	1.52e−15	2.86e+01	0.00e+00	5.80e-23
F2	Mean	8.87e+01	7.82e+01	1.09e+02	3.65e+02	4.66e+01	9.15e+01	1.61e+01
	Std	5.52e+00	4.25e+01	4.75e+00	9.02e+00	7.14e-01	7.14e-01	6.67e-01
F3	Mean	1.04e+03	1.33e+03	1.80e+03	9.10e+02	8.37e+05	8.98e+09	9.75e+02
	Std	5.14e+01	4.06e−01	3.37e−10	2.48e+01	4.39e+05	4.39e+08	4.32e+01
F4	Mean	5.89e+02	1.99e−01	1.37e-10	5.31e+03	7.58e+03	3.84e+03	5.50e+02
	Std	9.26e+00	4.56e−01	3.37e-10	2.48e+02	1.71e+02	1.71e+02	7.65e+01
F5	Mean	4.44e−16	1.18e−03	4.18e−13	3.94e−04	5.89e+00	0.00e+00	1.10e-16
	Std	0.00e+00	3.27e−03	2.78e−14	1.97e−03	3.91e−01	0.00e+00	0.00e+00
F6	Mean	3.44e−13	1.02e−12	1.06e−12	2.15e+01	1.89e+01	7.76e+00	1.11e-14
	Std	5.32e−15	1.68e−13	7.68e−14	2.19e+00	2.49e+00	8.92e−02	2.32e-15
F7	Mean	−1.30e+04	−1.43e+04	−1.47e+04	−1.25e+04	−6.62e+03	−7.51e+03	−3.00e+04
	Std	1.04e+02	8.27e+01	1.51e+01	9.36e+01	1.63e+01	1.64e+01	5.30e+00

Fig. 5 Comparison on CEC 2010

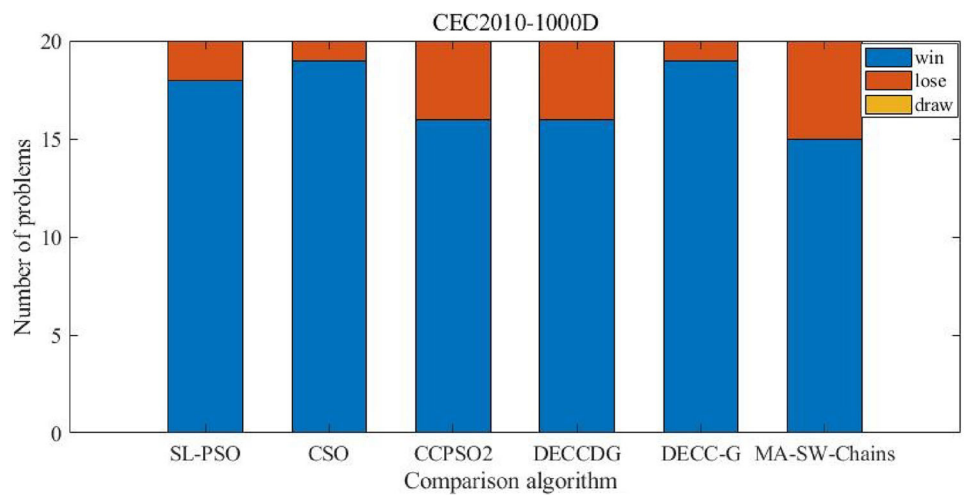


Fig. 6 Convergence tendency on F7 problem

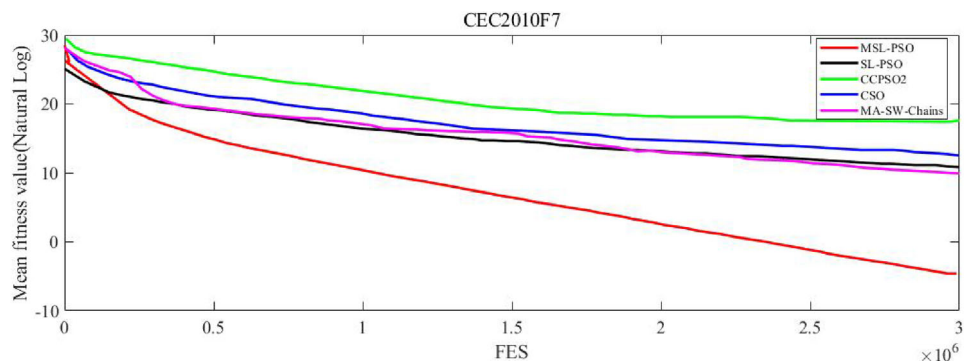
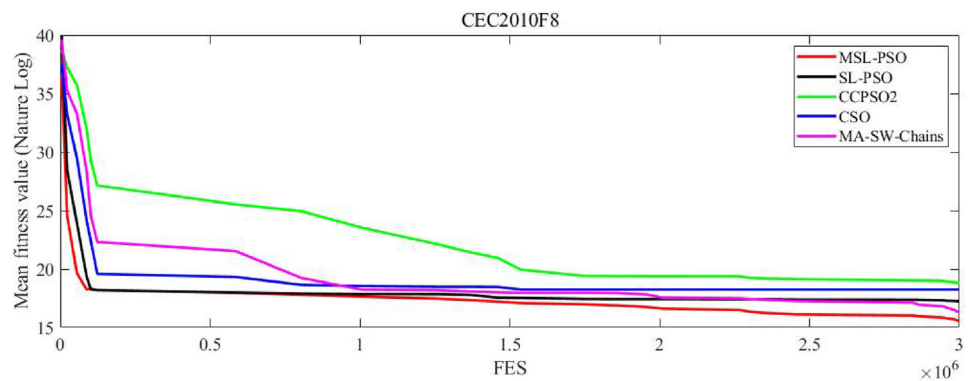


Table 7 Comparisons of the mean value and standard deviation between MSL-PSO and other algorithms on CEC2010 test problems with 1000 dimensions ($D = 1000$)

		SL-PSO	CSO	CCPSO2	DECCDG	DECC-G	MA-SW-Chains	MSL-PSO
F1	Mean	8.73e−18	4.75e−12	1.88e+00	7.80e+03	2.93e−07	2.10e−14	8.31e-19
	Std	3.30e−18	7.90e−13	2.26e+00	2.66e+04	8.62e−08	1.99e−14	1.43e-19
F2	Mean	1.93e+03	7.48e+03	5.06e+00	4.43e+03	1.31e+03	8.10e+02	7.92e+02
	Std	1.12e+02	2.63e+02	1.10e+00	1.98e+02	3.26e+02	5.88e+01	1.25e+01
F3	Mean	1.85e+00	2.57e−09	5.61e−03	1.67e+01	1.39e+00	7.28e−13	1.45e-13
	Std	3.30e−01	3.35e−10	1.73e−03	3.10e−01	9.73e−02	3.40e−13	8.44e-13
F4	Mean	3.04e+11	6.87e+11	2.14e+12	4.95e+12	1.70e+13	3.53e+11	5.30e+11
	Std	7.16e+10	1.79e+11	1.54e+12	1.33e+12	3.37e+12	3.12e+10	2.77e+12
F5	Mean	3.17e+07	2.46e+06	4.56e+08	1.49e+08	2.63e+08	1.68e+08	5.98e+06
	Std	6.21e+06	1.79e+06	1.38e+08	2.15e+07	6.84e+07	1.04e+08	3.43e+06
F6	Mean	2.15e+01	8.16e−07	1.79e+07	1.63e+01	4.96e+06	8.14e+04	9.07e-08
	Std	2.63e+00	2.68e−08	5.21e+06	3.45e−01	1.03e+06	2.84e+05	7.45e-09
F7	Mean	6.49e+04	2.13e+04	2.48e+08	1.14e+04	1.63e+08	1.03e+02	9.12e-02
	Std	5.60e+04	3.86e+03	4.31e+08	1.26e+04	5.44e+08	8.70+01	7.11e-01
F8	Mean	2.34e+07	3.86e+07	3.55e+07	3.30e+07	6.44e+07	1.41e+07	8.16e+06
	Std	2.64e+06	6.81e+04	3.30e+07	2.63e+07	2.64e+07	3.68e+07	9.47e+05
F9	Mean	3.22e+07	6.68e+07	1.00e+08	5.90e+08	3.21e+08	1.41e+07	1.23e+07
	Std	4.45e+06	5.73e+06	3.30e+07	6.45e+06	4.87e+06	1.15e+06	1.13e+06
F10	Mean	2.60e+03	9.58e+03	5.11e+03	4.55e+03	1.06e+04	2.07e+03	6.55e+03
	Std	2.17e+02	7.67e+01	7.81e+02	1.29e+02	3.11e+02	1.44e+02	9.85e+01
F11	Mean	2.30e+01	3.98e−08	1.98e+02	1.04e+01	2.34e+01	3.80e+01	5.83e-12
	Std	2.10e+00	5.12e−09	2.12e+00	8.71e−01	1.73e+00	7.35e+00	1.28e-13
F12	Mean	1.75e+04	4.37e+05	4.09e+04	2.56e+03	8.93e+04	3.62e-06	1.06e+04
	Std	9.70e+03	6.22e+04	1.19e+04	9.55e+03	1.08e+03	5.92e-07	8.61e+03
F13	Mean	8.48e+02	5.53e+02	1.32e+03	5.64e+03	5.12e+03	1.25e+03	4.72e+02
	Std	3.74e+02	2.32e+02	1.72e+02	4.16e+03	3.65e+03	5.72e+02	6.54e+01
F14	Mean	2.49e+08	2.46e+08	2.58e+08	3.40e+08	8.08e+08	3.11e+07	1.38e+07
	Std	1.53e+07	1.53e+07	1.19e+08	7.52e+07	2.42e+07	1.93e+06	5.94e+06
F15	Mean	1.01e+04	1.11e+04	1.05e+04	5.86e+03	1.22e+04	2.74e+03	7.06e+02
	Std	5.23e+01	8.65e+01	1.35e+03	8.24e+02	8.42e+02	1.22e+02	3.34e+01
F16	Mean	5.89e−08	5.68e−08	3.97e+02	7.57e-13	7.66e+01	9.98e+01	7.11e−12
	Std	5.61e−09	5.61e−09	5.73e−01	6.43e+00	6.43e+00	1.40e+01	1.14e−13
F17	Mean	2.20e+06	2.21e+06	1.32e+05	3.05e+04	2.87e+05	1.24e+00	5.56e+04
	Std	1.56e+05	1.55e+05	5.25e+04	2.24e+03	2.24e+04	1.25e-01	1.32e+04
F18	Mean	1.73e+03	1.64e+03	2.91e+03	1.46e+10	2.46e+04	1.30e+03	1.27e+03
	Std	5.22e+02	5.22e+02	2.45e+02	2.03e+09	1.53e+04	4.36e+02	6.32e+02
F19	Mean	1.01e+07	9.86e+06	1.53e+06	1.74e+06	1.11e+06	2.85e+05	8.01e+06
	Std	5.64e+05	5.64e+05	7.10e+04	1.10e+05	6.23e+04	1.78e+04	5.67e+05
F20	Mean	1.05e+03	1.07e+03	2.15e+03	6.28e+10	4.06e+03	1.07e+03	9.37e+03
	Std	1.49e+02	1.49e+02	1.79e+02	6.97e+09	1.05e+01	7.29e+01	1.57e+03

Fig. 7 Convergence tendency on F8 problem



pared on the performance, and the loser will learn from the winner and the winner be kept to next generation.

5. SL-PSO [6]: A new learning technique was proposed to solve large-scale optimization problems, in which the population is sorted in descending order, and each individual learns from its demonstrators who have better fitness values than this individual.
6. CCPSO2 [30]: It is a PSO variant based CC, in which the decision variables are randomly grouped, the size of which is also randomly generated.
7. sep-CMA-ES [47]: A simple covariance matrix adaptation evolution strategy variant proposed for large-scale optimization problems, in which the internal time is reduced and the space complexity is simplified from quadratic to linear.
8. EPUS-PSO [49]: A PSO variant is adopted to optimize the initial parameters of the Reservoir Computing. The results of EPUS-PSO are better than those obtained by an exhaustive search for global parameters generation of Reservoir Computing.
9. DMS-L-PSO [71]: The multi-swarm learning strategy is raised in DMS-L-PSO, and the sub-swarms will be re-grouped to exchange information among all the individuals.
10. MA-SW-Chains [35]: Each individual is assigned to local search intensity based on its features, and then, different local searches are chained.

Tables 4, 5, and 6 give the statistical results on CEC2008 test problems with 100, 500, and 1000 dimensions, respectively, and Fig. 4 is a bar-graph to show the number of CEC 2008 test problems with 100, 500, and 1000 dimensions that our proposed MSL-PSO obtained better, equal, and worse mean optimal results than each comparison algorithm. In Fig. 4, the region in red, red, and earth yellow represent the number of problems that MSL-PSO wins, loses, and draw with each comparison algorithm, respectively. From Tables 4, 5 and 6 and Fig. 4, we can see that MSL-PSO outperforms the other six excellent algorithms on most problems. To be specific, MSL-PSO obtains better results on 15/21 problems than

SL-PSO, CCPSO2, MLCC, sep-CMA-ES, and EPUS-PSO. Specially, the mean optimal solutions found by MSL-PSO are all better than EPUS-PSO. Note that the results of DMS-L-PSO come from [71], in which the maximum number of fitness evaluation on CEC2008 is $5000 * D$. Compared to DMS-L-PSO, we can easily find that our MSL-PSO algorithm obtained better results than DMS-L-PSO on all seven benchmark problems except F1 and F6 with 100, 500, and 1000 dimensions, respectively, even the number of fitness evaluations of DMS-L-PSO is much more than SML-PSO. From Tables 4, 5, and 6, we can see that the results of SML-PSO is not better than, but almost same to, those of DMS-L-PSO on F1 and F6, which, we think, is because a local search is utilized in the latter, so that the precise of the results can be improved. SML-PSO is better than the other compared algorithms on solving unimodal problems, i.e., F2, in CEC2008 test suite. Also, MSL-PSO gets better results on F7 than other algorithms. MSL-PSO algorithm failed to obtain better results than sep-CMA-ES and MLCC on F3 and F4, respectively; however, the result of MSL-PSO is still better than the other four compared algorithms.

Table 7 lists the results obtained by six state-of-the-art algorithms and our proposed MSL-PSO on 20 CEC2010 problems with 1000 dimensions, and Fig. 5 gives the bar-graph to show the number of CEC 2010 benchmark problems with 1000 dimensions that our MSL-PSO wins, loses, and draws with each comparison algorithm. Among the 20 test instances, MSL-PSO obtained 11 best mean optimal results among all of these algorithms. Compared to CCPSO2, DEC-CDG, and DECC-G, which utilize the cooperative coevolutionary strategies, we can see that the proposed MSL-PSO got only four worse results than both CCPSO2 and DEC-CDG, and only obtain one worse mean result than DECC-G. The results compared to SL-PSO and CSO, both of which are PSO variants, show that MSL-PSO can get 18/20 and 19/20 better mean optimal solutions than both of these algorithms, which shows that our proposed learning strategy is efficient to solve the large-scale optimization problems.

For further observations, Fig. 6 and Fig. 7 plot the convergence tendency of MSL-PSO, SL-PSO, CCPSO2, CSO, and

MA-SW-Chains on CEC2010 F7 and F8 problems, respectively, in which F7 is unimodal function and F8 is multimodal one. From Figs. 6 and 7, we can see that the MSL-PSO can converge much quicker than others, especially on unimodal F7. From Table 7, we can also find that MSL-PSO obtained worse results on the unimodal functions F2, F17, and F19, which, we analyze, is because all of these three problems are separable. Therefore, the cooperative coevolutionary algorithms are much suitable for solving this kind of problems.

Conclusion

A multiple-strategy learning particle swarm optimization was proposed for solving large-scale optimization problems. Some positions are probed for each individual by learning its demonstrators and the mean position of the population at first. And then, each individual updates its velocity and position by learning two demonstrators coming from different sub-sets, which is expected to balance the diversity and convergence. Experimental results show that MSL-PSO has a better performance on solving large-scale optimization problems proposed in CEC2008 and CEC2010. However, for the separable problems, its performance is not better than cooperative coevolutionary algorithms, so in the future, we will try to introduce the grouping approaches used in CC algorithms into our proposed MSL-PSO to get much better performance on solving this kind of problems.

Acknowledgements This work was supported in part by National Natural Science Foundation of China (Grant no. 61876123), Natural Science Foundation of Shanxi Province (201801D121131, 201901D111264, and 201901D111262), Shanxi Science and Technology Innovation project for Excellent Talents (201805D211028), Shanxi Province Science Foundation for Youths (201901D211237), the Doctoral Scientific Research Foundation of Taiyuan University of Science and Technology (20162029), and the China Scholarship Council (CSC).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Cagnina L, Errecalde M, Ingaramo D, Rosso P (2014) An efficient particle swarm optimization approach to cluster short texts. *Inf Sci* 265(5):36–49
- Cai Z, Peng Z (2002) Cooperative coevolutionary adaptive genetic algorithm in path planning of cooperative multi-mobile robot systems. *J Intell Robot Syst* 33(1):61–71
- Chen WN, Jia YH, Zhao F, Luo XN, Jia XD, Zhang J (2019) A cooperative co-evolutionary approach to large-scale multisource water distribution network optimization. *IEEE Trans Evolut Comput* 23(5):842–857
- Cheng R, Jin Y (2014) Demonstrator selection in a social learning particle swarm optimizer. In: *IEEE congress on evolutionary computation*
- Cheng R, Jin Y (2015) A competitive swarm optimizer for large scale optimization. *IEEE Trans Cybern* 45(2):191–204
- Cheng R, Jin Y (2015) A social learning particle swarm optimization algorithm for scalable optimization. *Inf Sci* 291:43–60
- Cheng R, Sun C, Jin Y (2013) A multi-swarm evolutionary framework based on a feedback mechanism. In: *IEEE congress on evolutionary computation*, pp 718–724. IEEE, New York
- Cuevas E, González A, Zaldívar D, Pérez-Cisneros M (2015) An optimisation algorithm based on the behaviour of locust swarms. *Int J Bio-Inspired Comput* 7(6):402–407
- De Falco I, Cioppa A.D, Trunfio G.A (2017) Large scale optimization of computationally expensive functions: an approach based on parallel cooperative coevolution and fitness metamodeling. In: *Proceedings of the genetic and evolutionary computation conference companion*, pp 1788–1795
- De Falco I, Della Cioppa A, Trunfio GA (2019) Investigating surrogate-assisted cooperative coevolution for large-scale global optimization. *Inf Sci* 482:1–26
- Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans Evolut Comput* 6(2):182–197
- Du ZG, Pan JS, Chu SC, Luo HJ, Hu P (2020) Quasi-affine transformation evolutionary algorithm with communication schemes for application of RSSI in wireless sensor networks. *IEEE Access* 8:8583–8594
- Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: *MHS'95. Proceedings of the sixth international symposium on micro machine and human science*, pp 39–43. IEEE, New York
- Fan Q, Yan X (2015) Self-adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies. *IEEE Trans Cybern* 46(1):219–232
- Gong YJ, Ge YF, Li JJ, Zhang J, Ip WH (2016) A splicing-driven memetic algorithm for reconstructing cross-cut shredded text documents. *Appl Soft Comput* 45:163–172
- Gong YJ, Zhang J, Chung SH, Chen WN, Zhan ZH, Li Y, Shi YH (2012) An efficient resource allocation scheme using particle swarm optimization. *IEEE Trans Evolut Comput* 16(6):801–816
- Hansen N, Auger A, Ros R, Finck S, Posik P (2010) Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In: *Genetic & evolutionary computation conference*
- Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. *Evolut Comput* 9(2):159–195
- Ho SY, Lin HS, Liauh WH, Ho SJ (2008) OPSO: orthogonal particle swarm optimization and its application to task assignment problems. *IEEE Trans Syst Man Cybern Part A Syst Hum* 38(2):288–298
- Hu M, Wu T, Weir JD (2013) An adaptive particle swarm optimization with multiple adaptive methods. *IEEE Trans Evolut Comput* 17(5):705–720
- Ishaque K, Salam Z (2013) A deterministic particle swarm optimization maximum power point tracker for photovoltaic system under partial shading condition. *IEEE Trans Ind Electron* 60(8):3195–3206

22. Jia YH, Chen WN, Gu T, Zhang H, Yuan HQ, Kwong S, Zhang J (2018) Distributed cooperative co-evolution with adaptive computing resource allocation for large scale optimization. *IEEE Trans Evolut Comput* 23(2):188–202
23. Jr IF, Perc M, Kamal S.M, Fister I (2015) A review of chaos-based firefly algorithms: perspectives and research challenges. *Appl Math Comput* 252(C):155–165
24. Kashan AH, Kashan MH, Karimiyan S (2013) A particle swarm optimizer for grouping problems. *Inf Sci* 252(17):81–95
25. Kazimipour B, Omidvar M.N, Li X, Qin A.K (2014) A novel hybridization of opposition-based learning and cooperative co-evolutionary for large-scale optimization. In: 2014 IEEE congress on evolutionary computation (CEC), pp 2833–2840. IEEE, New York
26. Kennedy J (2000) Stereotyping: improving particle swarm performance with cluster analysis. In: Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512), vol. 2, pp 1507–1512. IEEE, New York
27. Kennedy J, Eberhart R (1995) Particle swarm optimization (PSO). In: IEEE international conference on neural networks, pp 1942–1948
28. LaTorre A, Muelas S, Peña JM (2013) Large scale global optimization: Experimental results with MOS-based hybrid algorithms. In: IEEE congress on evolutionary computation, pp 2742–2749. IEEE, New York
29. Li X, Mei Y, Yao X, Omidvar MN (2014) Cooperative co-evolution with differential grouping for large scale optimization. *IEEE Trans Evolut Comput* 18(3):378–393
30. Li X, Yao X (2011) Cooperatively coevolving particle swarms for large scale optimization. *IEEE Trans Evolut Comput* 16(2):210–224
31. Liang JJ, Qin AK, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans Evolut Comput* 10(3):281–295
32. Liao T, Socha K, Oca MAMD, Stutzle T, Dorigo M (2014) Ant colony optimization for mixed-variable optimization problems. *IEEE Trans Evolut Comput* 18(4):503–518
33. Mei Y, Omidvar MN, Li X, Yao X (2016) A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. *ACM Trans Math Softw (TOMS)* 42(2):1–24
34. Molina D, LaTorre A, Herrera F (2018) SHADE with iterative local search for large-scale global optimization. In: 2018 IEEE congress on evolutionary computation (CEC), pp 1–8. IEEE, New York
35. Molina D, Lozano M, Herrera F (2010) MA-SW-chains: memetic algorithm based on local search chains for large scale continuous global optimization. In: IEEE congress on evolutionary computation, pp 1–8. IEEE, New York
36. Montalvo I, Izquierdo J, Pérez R, Iglesias PL (2008) A diversity-enriched variant of discrete PSO applied to the design of water distribution networks. *Eng Optim* 40(7):655–668
37. Omidvar MN, Li X, Mei Y, Yao X (2013) Cooperative co-evolution with differential grouping for large scale optimization. *IEEE Trans Evolut Comput* 18(3):378–393
38. Omidvar MN, Yang M, Mei Y, Li X, Yao X (2017) DG2: A faster and more accurate differential grouping for large-scale black-box optimization. *IEEE Trans Evolut Comput* 21(6):929–942
39. Palafox L, Noman N, Iba H (2013) Reverse engineering of gene regulatory networks using dissipative particle swarm optimization. *IEEE Trans Evolut Comput* 17(4):577–587
40. Pan JS, Hu P, Chu SC (2019) Novel parallel heterogeneous meta-heuristic and its communication strategies for the prediction of wind power. *Processes* 7(11):845
41. Pan JS, Kong L, Sung TW, Tsai PW, Snášel V (2018) A clustering scheme for wireless sensor networks based on genetic algorithm and dominating set. *J Internet Technol* 19(4):1111–1118
42. Potter MA, Jong KAD (1994) A cooperative coevolutionary approach to function optimization. *Third Parallel Probl Solving Form Nat* 866:249–257
43. Price K, Storn RM, Lampinen JA (2006) Differential evolution: a practical approach to global optimization. Springer, New York
44. Qiang Y, Chen WN, Yu Z, Gu T, Yun L, Zhang H, Zhang J (2017) Adaptive multimodal continuous ant colony optimization. *IEEE Trans Evolut Comput* 21(2):191–205
45. Ray T, Yao X (2009) A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning. In: 2009 IEEE congress on evolutionary computation, pp 983–989. IEEE, New York
46. Ren Z, Zhang A, Wen C, Feng Z (2013) A scatter learning particle swarm optimization algorithm for multimodal problems. *IEEE Trans Cybern* 44(7):1127–1140
47. Ros R, Hansen N (2008) A simple modification in CMA-ES achieving linear time and space complexity. In: International conference on parallel problem solving from nature, pp 296–305. Springer, New York
48. Ruiz-Cruz R, Sanchez EN, Ornelas-Tellez F, Loukianov AG, Harley RG (2013) Particle swarm optimization for discrete-time inverse optimal control of a doubly fed induction generator. *IEEE Trans Cybern* 43(6):1698–1709
49. Sergio AT, Ludermir TB (2012) PSO for reservoir computing optimization. In: International conference on artificial neural networks, pp 685–692. Springer, New York
50. Setayesh M, Zhang M, Johnston M (2013) A novel particle swarm optimisation approach to detecting continuous, thin and smooth edges in noisy images. *Inf Sci* 246:28–51
51. Shi Y, Teng H, Li Z (2005) Cooperative co-evolutionary differential evolution for function optimization. *Lect Notes Comput Sci* 3611:1080–1088
52. Stützle T (2009) Ant colony optimization. In: International conference on evolutionary multi-criterion optimization, pp 2–2. Springer, New York
53. Sun C, Ding J, Zeng J, Jin Y (2018) A fitness approximation assisted competitive swarm optimizer for large scale expensive optimization problems. *Memet Comput* 10(2):123–134
54. Sun C, Jin Y, Cheng R, Ding J, Zeng J (2017) Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems. *IEEE Trans Evolut Comput* 21(4):644–660
55. Sun Y, Kirley M, Halgamuge SK (2017) A recursive decomposition method for large scale continuous optimization. *IEEE Trans Evolut Comput* 22(5):647–661
56. Tian J, Sun C, Tan Y, Zeng J (2020) Granularity-based surrogate-assisted particle swarm optimization for high-dimensional expensive optimization. *Knowl Based Syst* 187:104815
57. Tian J, Tan Y, Zeng J, Sun C, Jin Y (2018) Multiobjective infill criterion driven gaussian process-assisted particle swarm optimization of high-dimensional expensive problems. *IEEE Trans Evolut Comput* 23(3):459–472
58. Tizhoosh HR (2005) Opposition-based learning: a new scheme for machine intelligence. In: International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06), vol 1, pp 695–701. IEEE, New York
59. Weise T, Chiong R (2012) Evolutionary optimization: pitfalls and booby traps. *J Comput Sci Technol* 27(5):907–936
60. Wu G, Shen X, Li H, Chen H, Lin A, Suganthan PN (2018) Ensemble of differential evolution variants. *Inf Sci* 423:172–186
61. Yang Q, Chen WN, Da Deng J, Li Y, Gu T, Zhang J (2017) A level-based learning swarm optimizer for large-scale optimization. *IEEE Trans Evolut Comput* 22(4):578–594

62. Yang Q, Chen WN, Gu T, Zhang H, Deng JD, Li Y, Zhang J (2016) Segment-based predominant learning swarm optimizer for large-scale optimization. *IEEE Trans Cybern* 47(9):2896–2910
63. Yang Q, Chen W.N, Gu T, Zhang H, Yuan H, Kwong S, Zhang J (2019) A distributed swarm optimizer with adaptive communication for large-scale optimization. *IEEE Trans Cybern*
64. Yang Q, Chen WN, Zhang J (2018) Evolution consistency based decomposition for cooperative coevolution. *IEEE Access* 6:51084–51097
65. Yang Z, Tang K, Yao X (2007) Differential evolution for high-dimensional function optimization. In: 2007 IEEE congress on evolutionary computation, pp 3523–3530. IEEE, New York
66. Yang Z, Tang K, Yao X (2008) Large scale evolutionary optimization using cooperative coevolution. *Inf Sci* 178(15):2985–2999
67. Yang Z, Tang K, Yao X (2008) Multilevel cooperative coevolution for large scale optimization. In: 2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence), pp 1663–1670. IEEE, New York
68. Yu H, Ying T, Zeng J, Sun C, Jin Y (2018) Surrogate-assisted hierarchical particle swarm optimization. *Inf Sci* 454–455
69. Yu Y, Yu X (2007) Cooperative coevolutionary genetic algorithm for digital IIR filter design. *IEEE Trans Ind Electron* 54(3):1311–1318
70. Zhang YH, Gong YJ, Zhang HX, Gu TL, Zhang J (2016) Toward fast niching evolutionary algorithms: a locality sensitive hashing-based approach. *IEEE Trans Evolut Comput* 21(3):347–362
71. Zhao SZ, Liang JJ, Suganthan PN, Tasgetiren MF (2008) Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. In: 2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence), pp 3845–3852. IEEE, New York
72. Zhu Z, Zhou J, Zhen J, Shi YH (2011) DNA sequence compression using adaptive particle swarm optimization-based memetic algorithm. *IEEE Trans Evolut Comput* 15(5):643–658

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.