Applied Soft Computing Journal 89 (2020) 106124

Contents lists available at ScienceDirect

Applied Soft Computing Journal

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

Multiple scale self-adaptive cooperation mutation strategy-based particle swarm optimization

Xinmin Tao *, Wenjie Guo, Qing Li, Chao Ren, Rui Liu

College of Engineering and Technology, Northeast Forestry University, Harbin, Heilongjiang 150040, China

ARTICLE INFO

Article history:
Received 8 January 2019
Received in revised form 14 January 2020
Accepted 21 January 2020
Available online 27 January 2020

Keywords:
Particle swarm optimization
Premature convergence
Multi-scale Gaussian mutations
Uniform mutation
Self-adaptive mutation threshold

ABSTRACT

Particle Swarm Optimization (PSO) algorithm has lately received great attention due to its powerful search capacity and simplicity in implementation. However, previous studies have demonstrated that PSO still suffers from two key drawbacks of premature convergence and slow convergence, especially when dealing with multi-modal optimization problems. In order to address these two issues, we propose a multiple scale self-adaptive cooperative mutation strategy-based particle swarm optimization algorithm (MSCPSO) in this paper. In the proposed approach, we adopt multi-scale Gaussian mutations with different standard deviations to promote the capacity of sufficiently searching the whole solution space. In the adopted multi-scale mutation strategy, large-scale mutation can make populations explore the global solution space and rapidly locate the better solution area at the early stage, thus avoiding the premature convergence and simultaneously speeding up the convergence, while small-scale mutation can allow the populations to more accurately exploit the local best solution area during the later stage, thus improving the accuracy of final solution. In order to guarantee the convergence speed while avoiding premature convergence, the standard deviations for multi-scale Gaussian mutations would be reduced with the increase of iterations, which can make populations pay more attention to local accurate solution exploitation during the later evolution stage and consequently speed up the convergence. In addition, the threshold for each dimension to execute mutation is also dynamically adjusted according to its previous mutation frequency, which can allow MSCPSO to better balance the global and local search capacities, thus avoiding premature convergence without reducing convergence speed. The extensive experimental results on various benchmark optimization problems demonstrate that the proposed approach is superior to other existing PSO techniques with good robustness.

1. Introduction

Particle swarm optimization (PSO) was originally proposed by Eberhart and Kennedy in 1995 [1]. As an optimization algorithm based on swarm intelligence [2], PSO searches for the optimal solution by imitating the social behavior of organisms such as birds. Like a bird, a candidate solution in PSO, sometimes called a particle, is generally composed of two key components: flying velocity and current position, which are dynamically adjusted according to its own flying historical information and the whole social swarm flying experience during the evolutionary process. Due to its simplicity in structure and implementation, PSO has not only shown good performance in various mathematics optimization fields including the target function optimization [3,4] and neural network training [5,6], and etc., but also has been widely applied in many real optimization problems such as manufacturing control in engineering optimization [7,8], multi-source scheduling in cloud computing [9,10], and other industry problems [11]. Unfortunately, like other classical swarm intelligence optimization algorithms, PSO still suffers from the problems of slow convergence and premature convergence, especially when dealing with the large-scale complex optimization problems.

To speed up the convergence of PSO, researchers began to pay more attention on parameter improvement, especially on the inertia weight. Inertia weight was firstly introduced into the velocity update equation by Shi and Eberhart [12], and afterward they further proposed a fuzzy adaptive method for the inertia weight adjustment in the modified PSO [13], which has been proven to greatly improve the performance of PSO. Based on their work, many variants of PSO algorithm have been proposed in the past few decades. For instance, Li N et al. [14] proposed a hybrid

* Correspondence to: College of Engineering and Technology, Northeast Forestry University, #26 Hexing Road, Xiangfang District, Harbin, Heilongjiang 150040, China.
E-mail addresses: taoxinmin@nefu.edu.cn (X. Tao), clockworkor@outlook.com (W. Guo), 812904921@qq.com (Q. Li), renchao@nefu.edu.cn (C. Ren), 1282369400@qq.com (R. Liu).

https://doi.org/10.1016/j.asoc.2020.106124
1568-4946/

particle swarm optimization (HPSOFW), which establishes a novel search behavior modal by incorporating fuzzy reasoning and a weighted particle, thus improving the searching capability of conventional PSO algorithm. Similarly, Nesamalar et al. [15] also used a fuzzy inference system to dynamically update the inertia weight. In addition, Wang et al. [16] proposed a self-adaptive learning based PSO (SLPSO) to adaptively select the most suitable inertia weights update strategies according to different stages of the evolutionary process. Different from the above methods adjusting the inertia weight, Chen et al. [17] shifted their focus to acceleration coefficients and proposed a hybrid particle swarm optimizer with sine cosine acceleration coefficients (H-PSO-SCAC) by adopting a sine map to adjust the acceleration coefficients. Although some advance has been made in improving convergence speed of PSO, high convergence speed especially in the early stage often tends to make the PSO algorithm prone to be trapped into local optimums and thus results in premature convergence.

In order to avoid premature convergence, researchers started to shift their concentration from parameters improvement to the population topology modification of PSO. Inspired by the study of Michalewicz [18] about population structure of evolutionary algorithms, Cesare et al. [19] used a stochastic Markov chain modal to define an intelligent topological structure of the swarm's population, in which the better particles have an important influence on the others. In addition, Wang et al. [20] also proposed a dynamic tournament topology strategy to improve PSO. In the proposed method, each particle is guided by several better solutions, which are randomly selected from the entire population. Although the selection of the better particles is stochastic, the reported experimental results demonstrated it still favors particles with better solutions. To keep the diversity of population and thus overcome the premature convergence, new population topology structures focusing more on local best particles for PSO were introduced in [21,22] and have been proved to work well in some cases. For the aforementioned PSO variants, although the swarm diversity can be maintained and the premature convergence is also avoided as reported, the intrinsic PSO evolutionary topology may be destroyed due to the introduction of complicated structure and thus leads to the reduction of the convergence speed. In order to address the dilemma, many researchers attempted to change the velocity update formula of PSO. For example, Liang et al. [23] attempted to use all other particles' historical best information to update a candidate particle's velocity via a learning strategy, which can allow the diversity of the swarm to be preserved and thus hinders premature convergence without reducing the convergence speed. Based on the same idea, Meng et al. [24] proposed the fully informed PSO (FIPSO), which also uses the information available from all the neighbors instead of the best one to guide a candidate particle evolution. Unlike the above proposed method, Cui et al. [25] utilized the predicted globally-optimal solution rather than all neighbor's information to adjust the velocity of a candidate particle. In addition to modifying the velocity update formula, some new learning strategies were also introduced to keep the balance between avoiding premature convergence and speeding up the convergence. For instance, Wang et al. [26] presented an enhanced PSO algorithm called GOPSO by combining the generalized opposition-based learning and Cauchy mutation. Moreover, Huang et al. [27] used an example set of multiple global best particles to update the positions of the particles, and proposed an example-based learning PSO (ELPSO) to balance the swarm diversity and convergence speed. Similarly, Cao et al. [28] also introduced a "worst replacement" strategy to update the swarm, where the position of the worst particle in the swarm is replaced by a better newly generated position, and the reported experimental results proved that the strategy is beneficial to the improvement of PSO convergence.

In recent years, some research demonstrated that integrating evolutionary mechanisms from other algorithms into PSO could also effectively improve the performance of PSO, such as simulated annealing algorithm (SA) and genetic algorithm (GA). As a case of integration with SA algorithm, Dong et al. [29] proposed a hybrid Particle swarm optimization algorithm to search a set of Pareto-optimal solutions, which employs the simulated annealing as a local search strategy to exploit the local solution space. Similarly, Wang et al. [30] adopted both SA algorithm and artificial neural network to enhance the global searching ability of PSO and thus developed a modified PSO, which was used to solve source estimation problems. In order to utilize some advantages of genetic algorithm, Tam et al. [31] integrated GA into PSO and presented a hybrid PSO method, which uses GA with two-point standard mutation and one-point refined mutation to further refine the exploitative search of PSO. Instead of directly combining two algorithms, some researchers attempted to introduce the mutation operation of the evolutionary algorithm into PSO to further strengthen its global space search ability and thus avoid premature convergence. For instance, Wei et al. [32] used polynomial mutation to maintain diversity in the external archive, which can effectively enhance the search capability of the algorithm and to prevent particles from falling into local optimum and premature convergence, but the convergence rate needs to be improved. Afterward, Chen et al. [33] also proposed a modified PSO algorithm with two differential mutations. Two adopted different differential mutation operations are assigned with two different control parameters, which can allow the particles in the top layer to sufficiently search global solution space. Similarly, Cheng et al. [34] further introduced the multi-dimensional uniform mutation operator to prevent algorithm from trapping into local optimum. In addition, Tong et al. [35] proposed a PSO optimization scale-transformation stochastic-resonance algorithm with stability mutation operator. Although this proposed method could promote the iteration speed and stability, the accuracy of its final solution should be further improved. Recently, Wang et al. [36] presented a hybrid PSO algorithm by employing an adaptive learning mutation strategy (ALPSO). Although ALPSO can obtain some good solutions in some cases as reported, the condition set for competitive learning mutation strategy used in ALPSO seems to be hard to be satisfied when swarm frequently gets trapped into local optima, which hence leads to the wastes of some evolutions and fitness evaluations, thus reducing the convergence speed. And Gholami et al. [37] presented a improved PSO algorithm by modifying personal best particle update strategy (MPBPSO) to schedule renewable generation in a micro-grid under load uncertainty and the experimental results demonstrated its effectiveness. In addition, Wang et al. [38] proposed a self-adaptive mutation differential evolution algorithm based on particle swarm optimization (DEPSO), which utilizes an improved DE mutation strategy with stronger global exploration ability and PSO mutation strategy with higher convergence ability to preserve its global solution searching capacity. In fact, although the above developed PSO algorithms can show better performance than traditional PSO on some cases due to the introduction of mutation, the single uniform mutation mechanism often fails to explore the better solution space for the population to escape from the local minimum especially when dealing with complex optimization problems, which can result in the waste of evolution iterations and even the failure of avoiding the premature convergence within the pre-specified finite iteration. In addition, the inappropriate required condition for mutation, which is set for balancing the PSO inherent evolution and additional mutation operation, can also be one key reason for no significant improvement in the performance of PSO.

In order to address the above limitations, we propose a multiple scale self-adaptive cooperative mutation strategy-based particle swarm optimization algorithm (MSCPSO) in this paper. In the proposed PSO algorithm, we adopted multi-scale Gaussian mutations with different standard deviations to guarantee the capacity of exploring better solution space and thus avoid premature convergence. Specifically, the large-scale mutation is responsible for exploring wider solution space region around a candidate particle, while the small-scale mutation is used to search local neighbor region around it. This mutation mechanism with multiple disperse scales can make the candidate particle more likely to find out the better solution than it and consequently help it escape from the local optimum. In order to ensure the convergence speed, standard deviations for multi-scale Gaussian mutations are dynamically set proportional to the fitness value of the whole population. Since the fitness value of the whole population would become smaller along with the iteration increasing when solving minimization problems, standard deviations for multi-scale Gaussian mutations would be reduced with the increase of iteration. These gradually decreasing standard deviations can make populations pay more attention to local accurate solution exploitation during the later evolution stage, which is favorable to speed up the convergence and improve the accuracy of final solution. Different from other PSO algorithms with mutations, the setting of mutation threshold condition and mutation operation are based on each dimension of solutions rather than the whole population or the global best solution. In addition, mutation threshold condition for each dimension is also self-adaptively adjusted according to its previous mutation frequency. This strategy can better balance the PSO inherent evolution and additional mutation operation such that the global search ability is maintained without losing the local search capacity. Note that the multi-scale mutation mechanism is similar to the idea of multiple subpopulations with different mutation strengths used in nested evolution strategies [2] to certain an extent since they both attempt to well balance the exploration and exploitation capabilities. However, the difference is that as mentioned above, the standard deviations for multi-scale Gaussian mutations are dynamically changed with the fitness value of the whole population, which can not only guarantee the capacity of exploring better solution space in the early stage but also ensure the convergence tendency in the later stage. In addition, the multi-scale mutation operations are dimensionally performed on the same particle satisfying pre-mature conditions to help it escape local optima with great chance rather than different scales for different particles, which is beneficial to keep a good tradeoff between PSO evolution strategy and multi-scale mutation strategy. The extensive experimental results on benchmark optimization problems show that the proposed algorithm outperforms other state-of-the-art modified PSO algorithms with statistical significance.

The remainder of this paper is organized as follows: A brief introduction of standard PSO is described in Section 2. Section 3 presents the details of Particle swarm optimization algorithm based on multiple scale self-adaptive cooperative mutation. In addition, we also analyze the mechanism of the proposed PSO algorithm and computational complexity. Section 4 shows the experimental results and discussions on extensive benchmark functions. Finally, Conclusions are drawn in Section 5.

2. Standard PSO algorithm

PSO is a swarm intelligence algorithm which was proposed by Eberhart and Kennedy [1] in 1995. Its main idea originated from imitation of the swarm behaviors in some ecosystems such as insects foraging and birds flying, which usually search for food or fly in a cooperative way. Specifically, each member of the swarm dynamically adjusts its searching or flying trajectory by learning its own experience and that of other members. Like an insect or a bird, a member of PSO swarm is also characterized by its velocity and position, and represents a candidate solution of the optimization problem, which is generally referred to as a particle. In order to implement coevolution, each particle dynamically adjusts its velocity according to its own current velocity inertia, its individual previous search experience and global search experience from the swarm during the evolutionary process. Then, each particle updates its position according to its corresponding adjusted velocity and its current position. For an optimization problem of n-dimension space, we let $\boldsymbol{x}_i(t) = (x_{i1}(t), x_{i2}(t), \ldots, x_{in}(t))^{\mathrm{T}}$ and $\boldsymbol{v}_i(t) = (v_{i1}(t), v_{i2}(t), \ldots, v_{in}(t))^{\mathrm{T}}$ denote the current position and current velocity of the ith candidate particle, respectively. Given a swarm composed of N particles in PSO, the detailed expressions of the velocity adjustment and the position updating of each particle are given as follows:

$$v_{id}(t+1) = \omega * v_{id}(t) + c_1 * r_1 * (p_{id}(t) - x_{id}(t)) + c_2 * r_2 * \left(p_{gd}(t) - x_{id}(t)\right) \quad (1)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (2)$$

where v_{id} denotes the current velocity of the ith candidate particle in the dth dimension, and x_{id} denotes the current position of the ith candidate particle in the dth dimension, $i \in 1, 2, \ldots, N; d \in 1, 2, \ldots, n$. $p_{id}(t)$ and $p_{gd}(t)$ refer to the dth dimensional values of the individual historical best solution and swarm historical best solution, respectively. c_1 and c_2 are two acceleration coefficients, which represent the learning weights from the individual previous search experience and global search experience, respectively. r_1 and r_2 are two random numbers , which are uniformly distributed in the range [0, 1], and are used to keep the diversity of the population. ω is the inertia weight concerning the current velocity, which can control the trade-off between the local and global search abilities.

As we can see from Eq. (1), the velocity adjustment of a candidate particle primarily depends on its own current velocity inertia and the shared information available from its individual historical best solution and swarm historical best solution. The velocity inertia of a candidate particle can make it focus on global solution space exploration, while individual historical best solution and swarm historical best solution can allow it to pay attention to the local accurate solution exploitation. On one hand, the enhancement of global solution space exploration can prevent a candidate particle from trapping into local optimum and thus avoid premature convergence. On the other hand, the promotion of local accurate solution exploitation can speed up the convergence and thus improve the accuracy of final solution. As previously described, premature convergence and slow convergence are two crucial issues faced by PSO. Therefore, how to control trade-off between the global and local search capacities is a key factor to improve the performance of PSO. In recent years, several attempts have been made and many variants of PSO have also been proposed. These algorithms attempted to balance the global and local search abilities through dynamically adjusting parameters including inertia weights and acceleration coefficients, modifying the population topology, and tuning the velocity updating rules etc. Although these variants of PSO can show better performance than standard PSO in some special cases as reported, they still cannot guarantee good diversity for the swarm to avoid premature convergence and simultaneously speed up the convergence, especially when dealing with complex multi-modal optimization problems.

To solve this dilemma, many researchers started to introduce the mutation operation of the evolutionary algorithm into PSO.

As reported, the additional mutation for those premature particles can make them possible to escape from the local optima during evolution process and thus avoid the premature convergence. However, the success of the modified PSO variants with mutation primarily depends on the efficient mutation strategy and appropriate mutation condition. In fact, inefficient mutation mechanism often results in the waste of evolution iterations and even the failure of avoiding the premature convergence within the pre-specified finite iteration. In addition, the inappropriate required condition for mutation also can lead to no significant improvement or even deterioration in the performance of PSO. Therefore, in order to adequately utilize additional mutation to improve the performance of PSO, designing an efficient mutation strategy and appropriate mutation condition has increasingly become the focus of attention.

3. Multiple scale cooperative mutation strategy-based PSO algorithm

In the aforementioned modified algorithms with mutation, the mutation operation applied in the premature particles aims to help them escape from the current trapped local optimum and thereby avoid premature convergence. Therefore, whether to successfully escape from the local optimum completely relies on the results of mutation operation. Suppose that the mutation results are evaluated by their fitness values, and then the good mutation result means that the mutated particle has better fitness value than its current one, which can allow it to successfully escape from the current local optimum and avoid premature convergence. By contrast, if the fitness value of the mutated particle is not better than the current one, this mutation operation is useless for it to escape from the local optimum. From the above analysis, we can conclude that the adequate mutation strategy is a crucial factor for the performance improvement of PSO with the help of mutation. However, in most of the modified PSOs with mutation, the applied mutation strategies are usually uniform mutation with single scale. The advantage of uniform mutation is its high demonstrated capability of global solution exploration with great variation. But, since we have no prior information concerning the solution space, it is very difficult to determine an appropriate scale for uniform mutation to help the premature particle escape from the local optimum. Specifically, if the scale is set large, we cannot guarantee that the fitness value of the mutated particle is better than that of the current one, especially during the late evaluation process when the better solution usually locates closely around the current solution. More specifically, uniform mutation with great scale can often lead to the mutated particle getting far away from the current one and thus locating in the area without better solution, indicating that the mutation operation is useless for the current particle to escape from local optimum. Hence, in order to alleviate this contradiction, we propose a multiple scale self-adaptive cooperative mutation strategy. In proposed mutation strategy, we not only apply Gaussian mutation operators with standard derivations of different scales to enhance the chance for the premature particles to escape from the local optimum, but also dynamically adjust these standard derivations with the increasing iterations, which can make the mutation focus on global solution space at the early stage while local solution space at the late stage. The detailed description about the multiple scale self-adaptive cooperative mutation strategy is presented in the following section.

3.1. Multiple scale cooperation mutation strategy

Suppose that the number of the scale is M and the size of the swarm is N, we first initialize the standard deviations of the multi-scale Gauss mutation operator:

$$\boldsymbol{\sigma}^{(0)} = \left(\sigma_1^{(0)}, \sigma_2^{(0)}, \ldots, \sigma_M^{(0)}\right) \tag{3}$$

Note that the each initial standard deviation is generally set within the range of optimization variables, and would be dynamically adjusted with the increasing of iteration during the evolutionary process. More specifically, at each iteration we first rank the particles in ascending order according to their corresponding fitness values when dealing with the minimization optimization problems, which means that the best candidate particles of swarm with the minimum fitness values would rand first. Next, we sequentially partition ranked particles into M subgroups and each sub-group contains $P = N/M$ particles. Let t denote the current iteration number, and then we calculate the average fitness value of each sub-group according to the following expression:

$$FitX_m^{(t)} = \sum_{p=1}^{P} \frac{f\left(\boldsymbol{x}_p^m\right)}{P}, m = 1, 2, \ldots, M \tag{4}$$

where $\boldsymbol{x}_p^m$ represents the pth particle which belongs to the mth subgroup and $FitX_m^{(t)}$ refers to the average fitness value of the mth subgroup at the tth iteration.

In order to enhance the diversity among standard deviations of multi-scale mutation, we dynamically adjust the standard deviation of each scale according to the average fitness value of its corresponding sub-group. The concrete expression for each scale standard deviation is given as follows:

$$\sigma_m^{(t)} = \sigma_m^{(t-1)} \exp\left(\frac{M * FitX_m^{(t)} - \sum_{m=1}^{M} FitX_m^{(t)}}{FitX_{max} - FitX_{min}}\right) \tag{5}$$

$$FitX_{max} = \max\left(FitX_1^{(t)}, FitX_2^{(t)}, \ldots, FitX_M^{(t)}\right) \tag{6}$$

$$FitX_{min} = \min\left(FitX_1^{(t)}, FitX_2^{(t)}, \ldots, FitX_M^{(t)}\right) \tag{7}$$

According to the above formula, we can find that the standard deviation of the scale corresponding to the top ranked subground would exhibit monotonically decreasing characteristics, while the standard deviation of the scale corresponding to the behind ranked sub-ground would exhibit monotonically increasing characteristics. Therefore, in order to avoid the standard deviation of the scale beyond scope of optimization variables, for the ones corresponding to the behind ranked sub-groups, we make additional regularization for these scales. The detailed procedure about the regularization is described in the following.

Procedure 1: The pseudo code of regularization procedure

1.**Input**: $\sigma_m^{(t)}$ and W, $[-\frac{W}{2}, \frac{W}{2}]$ is the range of optimization variables

2.**Output**: $\sigma_m^{(t)}$,

3.**While** $\sigma_m^{(t)} > W/2$

$$\sigma_m^{(t)} = \left|\frac{W}{2} - \sigma_m^{(t)}\right|$$

4.**End**

Through the above regularization, the standard deviations of the scales corresponding to the behind ranked sub-groups would behave fluctuation with the increasing of iterations and the standard deviation of the scale corresponding to the last ranked sub-group would fluctuate more frequently due to its value often

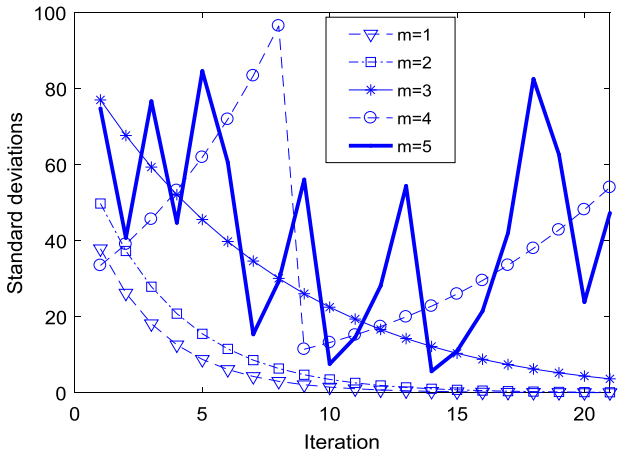

Fig. 1. The modification of standard deviations corresponding to different scales with the increasing of iteration when $M = 5$.

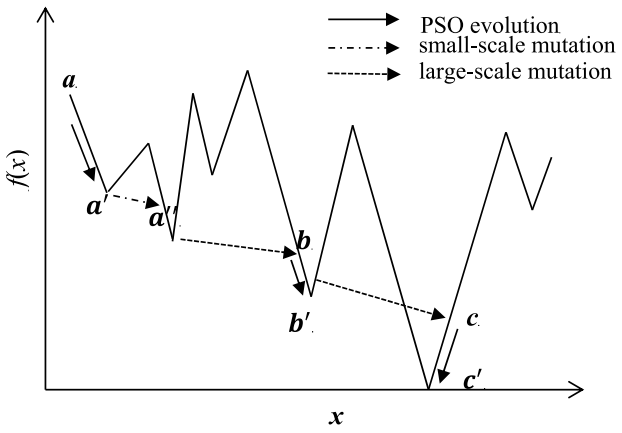

Fig. 2. Illustration for cooperative optimization mechanism.

beyond the scope. Therefore, regardless of the rank of the scale, the amplitude of variation for each scale standard deviation is also different from each other. In order to illustrate the diversity of multiple scale self-adaptive cooperation mutation strategy, we take 30-dimensional Tablet optimization problem as an example and set the number of scales to be $M = 5$ and the range of optimization variables be $[-100, 100]$. The modifications of standard deviations of different scales with the increasing of iteration are shown in Fig. 1.

As we can see from Fig. 1, the standard deviations of the scales corresponding to the top ranked sub-grounds decrease monotonically along with the increasing of iteration such as when $m = 1$, 2, and 3. Among those scales, the rates of decreasing are different from each other, which can allow the mutation to maintain the high diversity and thus enhance the searching capacity in the solution space, especially in local solution space. In addition, the property of monotonically decreasing with the increasing iteration can also guarantee that the mutation would pay more attention to the local accurate solution searching during the late evolution process, thus improving the accuracy of final solution. Unlike the aforementioned scales, the standard deviations of the scales corresponding to the behind ranked sub-grounds exhibit fluctuating characteristics with the growing iteration due to the effect of regularization. In addition, the fluctuation frequencies of those scales also differ from each other. Among them, the standard deviation of the scale $m = 5$ fluctuates more frequently than others. As described above, this is because that the standard deviation of the scale corresponding to the last ranked sub-group often exceeds the scope of the optimization variables and are more frequently regularized. In fact, those scales with different fluctuating frequencies can help the mutation thoroughly explore the whole solution space and thus enhance the capacity of searching in the global solution space. Finally, through the cooperation between the two types of mutation scales, the proposed mutation strategy can well-balance the global and local searching abilities, thus avoiding premature convergence and speed up the convergence.

3.2. Cooperative optimization mechanism of the proposed PSO algorithm

In order to conveniently illustrate the cooperative optimization mechanism of the proposed PSO, we assume that the minimization problem is a multi-modal optimization function which only includes one independent variable. The optimization process regarding this function by the proposed approach is shown in Fig. 2. Initially, we suppose that a candidate particle of the swarm locates in a. After several iterations, mainly updated by PSO inherent evolution strategy shown in Eq. (10), it evolves to the location a', which is one of local minimal solution of the multi-modal function. This update process is called 'climb down', which is responsible for the local accurate solution searching. Here, we assume that it cannot escape from this local optimum after several iterations and finally meets mutation condition. By multi-scale mutation, mainly small-scale one, it could find the better location a'' and thus escape from the current local minimal solution. Similar to the above evolutionary process except that the primary mutation is large-scale, it is assumed to arrive at the location b. After iteratively updated by PSO evolution, it is trapped into another local optimum b' again. Similar to the location a', it satisfies mutation condition after several useless iterations and undergoes the mutation operators. By multi-scale mutation, primarily large-scale one, it finds the better location c and subsequently moves to the location c at next iteration. Eventually, after PSO evolution update and possible small-scale mutation, it reaches the global optimal solution c'.

From the above illustration in Fig. 2, we can find that if the whole solution space is treated as a valley with many floors, the multi-scale mutation operation can help the particle located in valley floor to escape toward a better location. More specifically, the small-scale mutation is mainly applied to locate the better solution close to the current valley floor such as $a' \rightarrow a''$. On the contrary, the large-scale mutation operation can be used to search the better solution located far away from the current valley floor such as $b' \rightarrow c$, which is also beneficial to speed up the convergence. After a better region is found by multi-scale mutation such as b and c, the PSO evolution is responsible for searching the local region so as to find the more accurate solution $(a \rightarrow a', b \rightarrow b', c \rightarrow c')$, thus speeding up the convergence. This illustration can intuitively confirm that by the cooperative operation of PSO evolution, the small-scale and the large-scale mutations, the proposed algorithm can simultaneously maintain the global and the local searching capacities, and thus effectively avoid the premature convergence and speed up the convergence.

3.3. Self adaptive mutation condition for cooperation of local and global search capabilities

As described previously, an efficient mutation strategy and appropriate mutation condition are two crucial factors for successfully improving the performance of PSO via the introduction of mutation. Therefore, in addition to designing an efficient mutation strategy, how to set an appropriate mutation condition

is also a key research topic for the PSO variants with mutation to improve the performance. In fact, the improper mutation condition can result in the failure of performance improvement or even performance degradation. Specifically, if the threshold for mutation is too large, most of candidate particles can easily meet the mutation condition and thus undergo mutation, which can lead to some particles escape from the current evolutionary swarm. This escape would disorganize the inherent evolutionary structure of swarm and consequently make the algorithm unable to efficiently search the local solution space and thus improve the accuracy of final solution. On the contrary, if the threshold for mutation is too small, it will take a long time for the premature particles to meet mutation condition, which means that they may fail to escape from the local optimum during a pre-specified finite number of iterations and thus avoid the premature convergence. In order to address this contradiction, we develop a self-adaptive mutation threshold setting method. Unlike the other existing PSO variants with mutation, we perform mutation on each dimension of the candidate particle rather than the entire particle. Hence, we assume that each dimension of the particle is independent from each other, and self-adaptively set different mutation thresholds for each dimension. Specifically, for a given dimension, first we randomly initialize a relatively large mutation threshold for it and record the total number of the particles satisfying mutation condition regarding it. When the total number reaches a certain value, which indicates that there exist many particles satisfying the current mutation threshold regarding it, we dynamically adjust the current threshold according to the pre-specified ratio. The concrete procedure is described in the following.

$$G_d(t) = G_d(t-1) + \sum_{i=1}^{N} c_{id}(t) \tag{8}$$

$$\text{in which} \quad c_{id}(t) = \begin{cases} 0 & v_{id}(t) > T_d \\ 1 & v_{id}(t) < T_d \end{cases}$$

$$\text{If } G_d(t) > k_1 \text{ then}$$

$$G_d(t) = 0;\ T_d = \frac{T_d}{k_2} \tag{9}$$

where T_d denotes the mutation threshold regarding the dth dimension, and $G_d(t)$ denotes the total number of the particles satisfying mutation condition T_d. k_1 is a pre-specified threshold for $G_d(t)$, and k_2 is another constant that is used to control the decreased ratio of the threshold. N is the size of the swarm. According to the given the mutation threshold for each dimension, here we also present the mutation procedure for the minimization optimization problems in the following.

Procedure 2: The pseudo code of mutation operation procedure

Input: a candidate particle $\boldsymbol{x}_i(t), \boldsymbol{v}_i(t)$;
Output: $\boldsymbol{x}_i(t), \boldsymbol{v}_i(t)$;
$\boldsymbol{tv}_i^{(m)}(t) = \boldsymbol{v}_i(t), i = 1,2,\cdots,M+1$;
for each dimension d from 1 to D **do**
  if $v_{id}(t) < T_d$ **then**
    for each mutation scale m from 1 to M do
      $tv_{id}^{(m)}(t) = \sigma_m^{(t)} \times randn;$
    end for
    $tv_{id}^{(M+1)}(t) = V_{max} \times rand;$
  end if
end for
$f\left(\boldsymbol{x}_i(t) + \boldsymbol{tv}_i^{(j)}(t)\right) = \min_{1\le m\le M+1} f\left(\boldsymbol{x}_i(t) + \boldsymbol{tv}_i^{(m)}(t)\right);$
Output: $\boldsymbol{x}_i(t) = \boldsymbol{x}_i(t) + \boldsymbol{tv}_i^{(j)}(t); \boldsymbol{v}_i(t) = \boldsymbol{tv}_i^{(j)}(t);$

Where V_{max} is the upper bound of velocity, generally equal to $\frac{1}{2}|W|$, and f denotes the fitness function. Note that in order to further ensure the searching ability in global solution space, we also add uniform mutation in the procedure.

Moreover, due to the introduction of multiple scale self-adaptive cooperative mutation strategy, which can sufficiently guarantee efficient global searching ability, in the proposed algorithm we omit the weight inertia component in the velocity update formula to make PSO focus more on the local solution space searching and thus speed up the convergence. The applied velocity update formula without the weight inertia is given as follows:

$$\begin{aligned} v_{id}(t+1) &= c_1 * r_1 * (p_{id}(t) - x_{id}(t)) \\ &\quad + c_2 * r_2 * \left(p_{gd}(t) - x_{id}(t)\right) \end{aligned} \tag{10}$$

where the definitions of all variables are the same as in Eq. (1).

3.4. The framework of the proposed MSCPSO algorithm

In this section, we present a detailed framework of the proposed multiple scale self-adaptive cooperation mutation strategy-based PSO algorithm in the following,

Procedure 3: The Framework for the proposed algorithm

1: **Initialize**: $t = 1, k_1, k_2$, all particles $\boldsymbol{x}_i(t)$, and all velocities $\boldsymbol{v}_i(t)$, $\boldsymbol{\sigma}^{(0)}, V_{max}$ according to the range of optimization variables W, T is the maximal iteration number;
2: **Evaluate** the fitness value of each particle $f(\boldsymbol{x}_i(t))$;
3: **Update** the global and individual historical best solution $\boldsymbol{p}_g(t)$ and $\boldsymbol{p}_i(t)$;
4: **while** (other termination condition is not met) and $t < T$ **do**
4.1: **Update** all particles $\boldsymbol{x}_i(t)$ and all velocities $\boldsymbol{v}_i(t)$ according to Eq.(10);
4.2: **Apply** the mutation procedure 2 to execute mutation for all particles;
4.3: **Update** the standard deviation of multi-scale mutation according to Eqs. (5)-(7).
4.4 **Compute** the total number of the particles satisfying mutation condition according to Eq.(8);
4.5: **Update** the corresponding mutation threshold for each dimension according to Eq.(9);
4.6: **Update** $\boldsymbol{p}_g(t)$ and $\boldsymbol{p}_i(t)$;
end while

3.5. Computational complexity of MSCPSO

Compared with the standard PSO, the main difference of MSCPSO is the if condition statement (4.2–4.5 of Procedure 3) which was used to execute mutation for mutation condition-satisfied particles. Suppose that N is the size of the swarm and the solution dimension is D, the computational complexity of mutation execution is $O(XD)$, where X represents the number of particles which satisfy the mutation condition and $X \le N$. The computational complexity of the updating operation of the standard deviation of multi-scale mutation is $O(M)$, where M denotes the number of sub-groups. As a result, if the stop condition of MSCPSO is a fixed iteration number T, the overall computational complexity of MSCPSO is $O(NDT) + O(XDT) + O(MT)$. According to the above analysis, we can find that the proposed algorithm has slightly higher computational complexity than standard PSO for each iteration, but generally speaking, MSCPSO can often find the satisfactory solution just within much fewer iterations than standard PSO. This advantage makes it save many iterations and thus lessens the computational complexity, which has been proven in the subsequent experiments designed in Section 4.

4. Experimental results

4.1. Benchmark functions and experimental configurations

In order to demonstrate the effectiveness of the proposed PSO algorithm to avoid the premature convergence and speed up the convergence, we firstly carried out two groups of optimization experiments on 2 basic benchmark unimodal functions and 3

Table 1
The detailed information about 5 basic benchmark functions.

Quadric	$f_2(x) = \sum_{i=1}^{D}(\sum_{j=1}^{i} x_j)^2$	$[-100, 100]$	[0, 0, 0, …, 0]/0
Bent Cigar	$f_7(x) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2$	$[-100, 100]$	[0, 0, 0, …, 0]/0
Dminima	$f_{13}(x) = 78.332331408 + \sum_{i=1}^{D} \frac{x_i^4 - 16x_i^2 + 5x_i}{D}$	$[-5.12, 5.12]$	[0, 0, 0, …, 0]/0
Griewank	$f_{14}(x) = \sum_{i=1}^{D} x_i^2/4000 - \prod_{i=1}^{D} \cos(x_i/\sqrt{i}) + 1$	$[-600, 600]$	[0, 0, 0, …, 0]/0
Schwefels	$f_{19}(x) = 418.982887273 \times D - \sum_{i=1}^{D} x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-500, 500]$	[420.96, 420.96, 420.96, …, 420.96]/0

Table 2
The average number of evaluations and running time for 5 basic functions.

Function	Threshold	Algorithm	Total	Success	The average number of evaluations	Average time (s)
Quadric	1e-04	PSO	20	11	1.5669e+05	1.49
		MSCPSO	20	17	20624	0.12
BentCigar	1e-10	PSO	20	17	1.3675e+05	0.81
		MSCPSO	20	17	9576	0.06
Dminima	12	PSO	20	9	37462	0.45
		MSCPSO	20	20	200	0.02
Griewank	1e-02	PSO	20	9	58485	0.58
		MSCPSO	20	20	20260	0.19
Schwefels	9.2e+03	PSO	20	15	27709	0.26
		MSCPSO	20	20	370	0.02

basic benchmark multimodal functions, respectively, which are widely used as optimization test functions by most of the literatures. The detailed information regarding these used functions is summarized in Table 1 and all of those functions have only a minimization optimum. In this case, the performance of the proposed algorithm was evaluated and compared with the following popular PSO variants including conventional PSO, Accelerating particle swarm optimization using crisscross search presented in [24] (CSPSO), Globally-optimal Prediction-based Adaptive Mutation Particle Swarm Optimization proposed in [25] (GPAMPSO), PSO Optimization Scale-Transformation Stochastic-Resonance Algorithm With Stability Mutation Operator developed in [35] (STSRPSO) and other two PSO variants with mutation. The two PSO algorithms with mutation are more related to the proposed PSO algorithms, which include Particle Swarm Optimizer with two differential mutations (TDMPSO) proposed in [33] and Hybrid Particle Swarm Optimization Algorithm Using Adaptive Learning Strategy (ALPSO) which is introduced in [36]. In addition, we also added two representative PSO algorithms based on hybrid improved strategy proposed recently: MPBPSO [37] and DEPSO [38], and one successful CMA-ES improved algorithm: Bi-Pop-CMA-ES [39] into comparison. Bi-Pop-CMA-ES is compared because it has become currently one of the most outstanding algorithms for single-objective continuous optimization. BI-population CMA-ES presented by Hansen et al. [39] is a multi-start CMA-ES with equal budgets for two interlaced restart strategies (BI-Pop-CMA-ES), one with an increasing population size and the other with varying small population size and the experimental results show that BI-Pop-CMA-ES can efficiently solve most of the noisy functions of BBOB2009. The detailed parameters configurations regarding these compared algorithms are determined according to their corresponding references and are listed in Table 3. For CSPSO, pv represents the crossover probability to control how many dimensions in the population participate in the arithmetical crossover operation. c_3 is the developmental learning factor of GPAMPSO which is introduced to integrate the guiding effect of the predicted globally-optimal solutions into the velocity updating formula. b, R, and SNR refer to the stability threshold parameter, the frequency compression ratio, and the modified fitness maximum value of STSRPSO, respectively. For TDMPSO and DEPSO, F denotes the positive control parameter which can be used to find a better combination of mutations and $CR1$, $CR2$ are another two control parameters which can determine the fraction of vector components inherited from the mutant vector. $Prob_{Candidate}$ represents the possibility for a candidate particle to perform mutation operator in ALPSO. And for MPBPSO, $\varnothing_1$ and $\varnothing_2$ were used to control the balance between global and local search ability. In addition, in this case, the dimension of all benchmark functions is uniformly set to be 30 and each optimization variable corresponding to each dimension has the identical range for the convenience of implementation. The swarm size N for each PSO algorithm is set as 40. The running environments include windows 7 64 bit operation system with Intel i7 4.3 GHz, 4G memory and all PSO variants are implemented with Matlab R2010b.

4.2. Running time test and diversity analysis

In order to prove that the proposed algorithm can overcome the drawback of higher computational complexity at each iteration than traditional PSO and converge to the best solution more rapidly, we set the appropriate solution thresholds as termination conditions for 5 benchmark functions and the maximum number of evaluations is set to be 300 000. Each algorithm was run independently over 20 times for all benchmark functions to avoid the effect of randomness and the final results averaged over all successfully satisfied solution threshold termination condition run within the maximum number of evaluations are shown in Table 2. As shown in Table 2, we can find that although the proposed algorithm has high computational complexity as we analyzed in Section 3, its' average running time for each function is shorter than that of standard PSO since the iterations it needs to satisfy termination condition are much fewer for all functions. In addition, from the results, we also find that the proposed algorithm can successfully reach the threshold solutions more times than standard PSO for all benchmark functions. This is mainly because that multiple scale cooperative mutations strategy can allow the

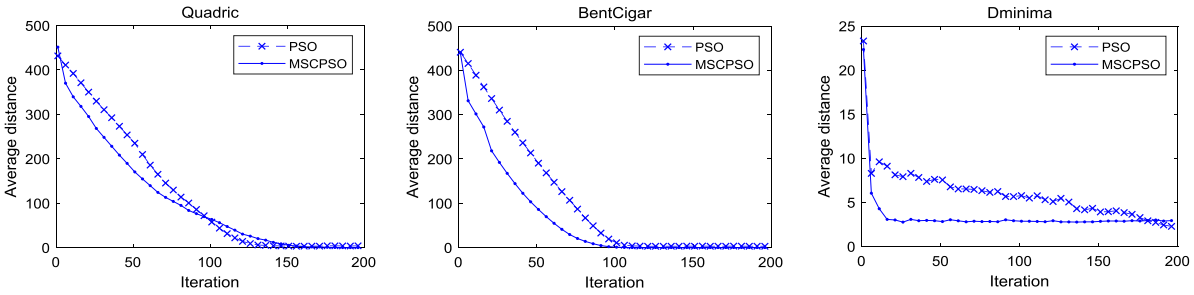

Fig. 3. Diversity plot of MSCPSO and PSO.

Table 3
The detailed parameters configurations regarding these compared PSO algorithms.

Algorithms	Inertia weights and Acceleration coefficients	Other parameters
PSO	$\omega = 0.4\, c_1 = 2, c_2 = 2$	
CSPSO	$\omega = 0.4\, c_1 = 2, c_2 = 2$	$pv = 0.8$
GPAMPSO	$\omega = 0.4\, c_1 = 0.9, c_2 = 0.9$	$c_3 = 0.4$
STSRPSO	$\omega = [0.4, 0.95]\, c_1 = 2, c_2 = 2$	$b = 1.7438, R = 66.2561$ SNR $= -15.4354$
TDMPSO	$\omega = 0.7298\, c_1 = c_2 = 1.49618$	$CR1 = 0.025, CR2 = 0.9, F = 0.5$
ALPSO	$\omega = [0.4, 0.9]\ c_1 = 2, c_2 = 2$	$Prob_{Candidate} = 0.5$
MPBPSO	$\omega = [0.4, 0.9]\, c_1 = 1, c_2 = 2$	$\varnothing_1 = 0.6, \varnothing_2 = 0.4$
DEPSO	$\omega = [0.4, 0.9]\, c_1 = 2, c_2 = 2$	$\gamma = 0.001,\ CR1 = 0.3, CR2 = 1, F = [0.1, 0.8]$
BiPop-CMA-ES		$\mu = \left\lfloor \frac{\lambda}{2} \right\rfloor, \omega_i = \frac{\ln(\mu+1) - lni}{\sum_{j=1}^{\mu}(\ln(\mu+1) - lnj)}\ \mu_W^{-1} = \sum_{i=1}^{\mu} \omega_i^2$ $d_\sigma = 1 + c_\sigma + 2\max(0, \sqrt{\frac{\mu_W - 1}{D+1}} - 1)$
MSCPSO	$c_1 = 1.4, c_2 = 1.4$	$k_1 = 5, k_2 = 10, M = 5$

proposed algorithm to rapidly locate the relatively good solution area and thus takes much less evaluations to satisfy solution threshold termination condition. Thus, although the computational complexity of the proposed method in each iteration is slightly higher than that of traditional PSO, the overall running time is less than that of traditional PSO due to the effect of fewer used number of evaluations.

Subsequently, to further show how exploration and exploitation is changed compare with the standard PSO, the diversity plots of 3 functions (Quadric, Rosenbrock, Bentcigar, Dminima) were given in Fig. 3, which shows the average distance between all solutions in each iteration, and each algorithm is still run independently over 20 times. As we can see in Fig. 3, when dealing with the unimodal functions, the proposed algorithm can converge rapidly to a relatively satisfactory search space in the early stage due to the effect of large-scale mutation, and in the later stage the descending will slow down gradually since the small-scale mutation can allow it to continue searching the local space more accurately. Similarly, the proposed method can still converge quickly in the early stage when dealing with mulit-modal function, and the multi-scale mutation strategy can make the proposed method preserve the diversity, which can enable it to search a better solution in the later stage, thus avoiding premature convergence.

4.3. Experimental comparison and analysis on 5 basic benchmark functions

In order to evaluate the accuracy of the obtained solution by different compared algorithms, we set uniformly the maximum generation of evaluations to be 300 000 and used it as only a termination condition for all algorithms. In addition, to avoid the effect of randomness, each algorithm is run independently over 30 times for all benchmark functions and then we calculated their statistical information concerning the final fitness values including maximum, minimum, mean, and standard deviation, which is shown in Table 5 (provided in Appendix A). The bold values denote the best results for each optimization function among all compared algorithms.

As has been shown in Table 5, we can find that the proposed PSO algorithm performs best on nearly all unimodal optimization functions except Quadric. Specifically, from the min values concerning the final fitness values shown in Table 5, we can find that for all unimodal optimization functions except for Griewank one, all other compared PSO algorithms cannot obtain the global minimal optimum among all runs. Compared to other PSO algorithms, the proposed PSO algorithm can obtain the global minimal optimum each run for all unimodal benchmark functions except Quadric and Bent Cigar. For Quadric and Bent Cigar benchmark functions, the mean fitness values obtained by the proposed PSO algorithm are 2.4503e−05 and 1.0593e−17, respectively. However, the min values obtained by the proposed PSO algorithm are all 0, which means that the proposed PSO algorithm can obtain the global minimal optimum at least one time among all runs. In addition, from the Max column shown in Table 5, we can further find that for Quadric and Bent Cigar, the maximum fitness values obtained by the proposed PSO algorithm among all runs are 3.3067e−04 and 3.1778e−16, respectively, which is the main reason for the increase of the means and standard deviation value. In fact, for Quadric and Bent Cigar benchmark functions, the proposed PSO algorithm obtains the global solution 3 and 11 times respectively among 30 runs, which indicates that the performance of the proposed PSO algorithm can be greatly improved by appropriately adjusting its parameters. In addition, for all multimodal benchmark functions, the proposed PSO algorithm can also achieve the better results compared to other PSO algorithm as shown in Table 5. Due to the existence of many local optimums confusing the searching process for global optimum,

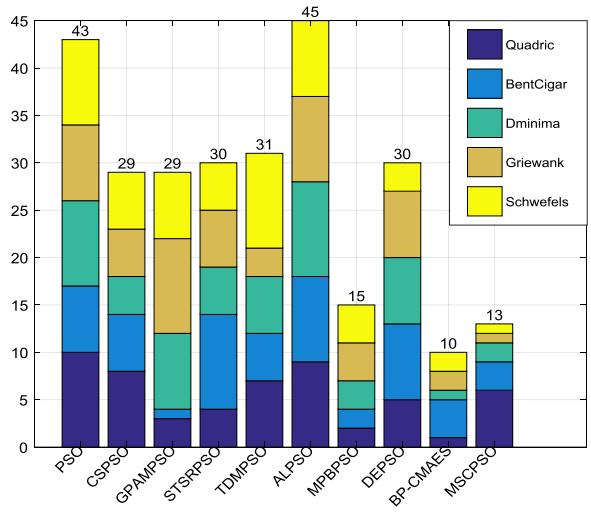

Fig. 4. Rank comparison of ten algorithms in 5 simple test functions.

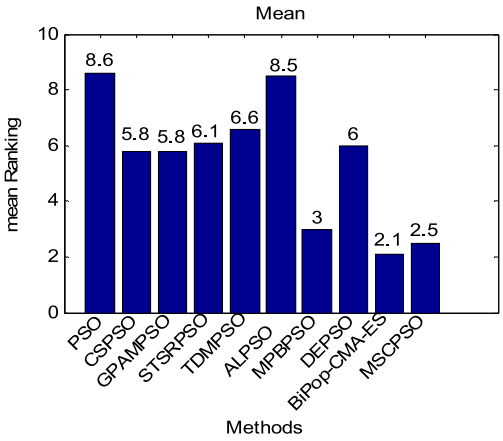

Fig. 5. Mean ranking of 5 simple test functions of different algorithms in terms of mean metric.

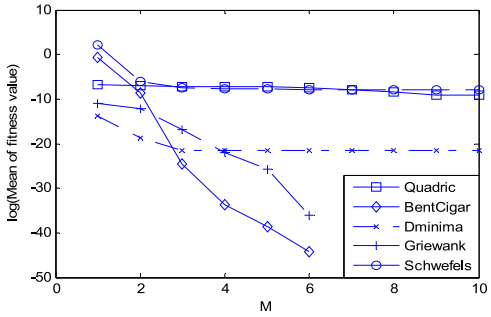

Fig. 6. The change of final solution accuracy with respect to different M values.

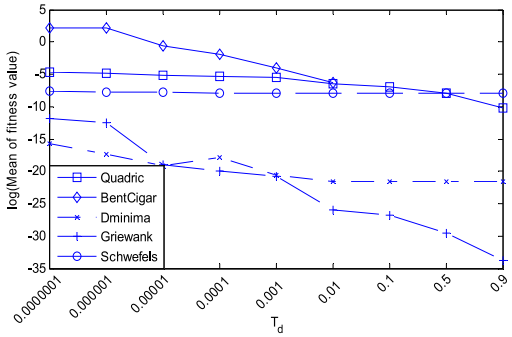

Fig. 7. The change of final solution accuracy with respect to different T_d values.

all other PSO variants fail to reach the global optimal solution and tend to be trapped into the local optimum during the fixed maximum generation of evaluations. Unlike other PSO variants, Bi-Pop-CMA-ES can obtain more favorable results regardless of on unimodal and multimodal functions, which can further demonstrate its outstanding advantage in addressing single-objective continuous optimization. Compared to Bi-Pop-CMA-ES, our proposed PSO algorithm can achieve comparative performances on the tested benchmark functions due to the application of multi scale mutation strategy. In order to make the comparison more intuitively and concisely, we also compare the rank regarding mean value obtained by each compared algorithm in each of the 5 optimization functions separately, which is shown in Fig. 4. For each benchmark function, the top-ranked algorithm will add 1 on the corresponding bar and the last-ranked algorithm will add 10, which means that the best algorithm would correspond to the lowest bar. The result shows that when dealing with these basic optimization problems, our proposed algorithm only performs worse than BiPop-CMA-ES.

In order to intuitively compare results, the mean ranks regarding Mean results of all compared algorithms on 5 basic functions are presented in Fig. 5. For each tested benchmark function, the algorithm with the best performance will be assigned a mean ranking of 1 while the worst algorithm will be assigned a mean ranking of 10. As shown in Fig. 5, BiPop-CMA-ES ranks first followed by the proposed PSO algorithm and MPBPSO.

4.4. Influence of parameters on the performance of the proposed PSO algorithm

In the proposed PSO algorithm, there are two important parameters to be pre-specified, which include the number of mutation scale M and mutation threshold regarding the dth dimension T_d. In order to investigate the effect of different M and T_d values on the accuracy of the final solution obtained by the proposed approach, we conducted the optimization experiments on five more difficult benchmark functions selected from the ones listed in Table 1. In order to eliminate the randomness, we perform the proposed PSO algorithm on each function over 30 runs for each parameter setting scenario and then calculate their average value of the final solutions.

4.4.1. Influences of M on the performance

In order to discuss the impact of M on the accuracy of the final solution obtained by the proposed PSO algorithm, we carried out the minimization optimization experiments on the selected five functions using the proposed PSO algorithm with different M values ranged from 1 to 10. The averaged fitness values obtained by 30 runs under different M values on each function are shown in Fig. 6. Note that when discussing the effect of M, other parameter settings are similar to before. For the convenience of comparison and analysis, we use log function to alleviate different magnitude effect on the final values of different functions obtained by the proposed PSO algorithms.

As has been shown in Fig. 6, it can be seen that the average fitness values of all five functions decrease with the increasing of M at the initial stage, such as $M = [1, \ldots, 6]$, while when M is further increased, such as $= [7, \ldots, 10]$, the average fitness values do not change significantly for all functions. Since the

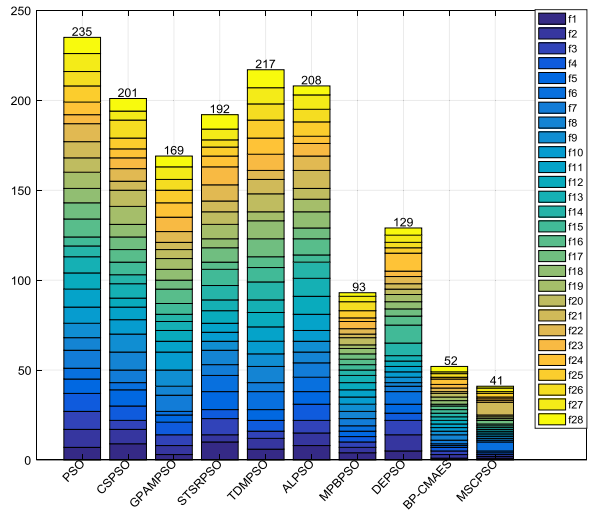

Fig. 8. Rank comparison of ten algorithms in 28 benchmark functions of CEC'2013.

lower fitness value means the better solution accuracy when dealing with minimization optimization problems, the experimental results indicates that the accuracy of the final solution obtained by the proposed PSO algorithm do not always become increasingly better with the increasing of M and can be the best when M is moderately set, such as $M = [5, 6]$. This is primarily because that the smaller M might lead to insufficient exploration on the global solution space at the early stage and thus make the populations possible to be trapped into the local optimum. In addition, the smaller M cannot implement the exploitation for the more accurate solution at the later stage during evolution due to the lack of sufficient mutation scales. On the other hand, the larger M is not essential for the improvement of the final solution accuracy since the multi-scale mutation strategy of the proposed PSO algorithm is designed just for single an optimal variable. Hence, we can conclude that M is usually set to be 5 or 6 for our proposed PSO algorithm to achieve good performance without considering the different dimensions.

4.4.2. Influences of T_d on the performance

The averaged fitness values of all five functions with respect to different T_d values ranged from 1e−7 to 0.9 are shown in Fig. 7. It can be found that the averaged fitness values of nearly all functions always become more accurate with the increasing of T_d values. Specifically, except Schwefels function, the accuracy of the averaged fitness values obtained by the proposed PSO algorithm on all other functions increases significantly when T_d is increased. This is mainly because that when the initial threshold for mutation T_d is set to relatively small, all particles have little chance to undergo mutation operation and thus fail to sufficiently explore the global solution space especially at the early stage during evolution, which can make the populations easy to be trapped in the local optimum. On the contrary, when the initial threshold T_d is set to relatively large, the particles have more chance to participate in mutation and thus enlarge the possibility of locating the global best solution area, which can effectively avoid the premature convergence at the early stage. In addition, due to the effect of the self-adaptive adjustment on the threshold for mutation, the T_d value would become smaller with the iterations, which can guarantee the PSO inherent evolution to exploit the more accurate solution space especially during the later evolutionary stage, thus well balancing global exploration and local exploitation. However, the larger T_d also indicates more time cost for computation. Therefore, in order to reasonably take advantage of multi-scale mutation strategy and simultaneously prevent from increasing the computational complexity, the initial threshold for mutation should be set to [0.1, 0.9] for superior performance.

Table 4
Wilcoxon signed ranks test of 28 benchmark functions.

Pairs of algorithm	R+	R−	n+	n−	ties	p-value
MSCPSO vs PSO	0	3.23e+05	27	1	0	2.66e−04
MSCPSO vs CSPSO	1.1400	1.57e+05	27	1	0	3.42e−04
MSCPSO vs GPAMPSO	1.1500	1.54e+05	27	1	0	5.28e−04
MSCPSO vs STSRPSO	1.1000	2.83e+05	27	1	0	3.29e−04
MSCPSO vs TDMPSO	0	2.47e+05	27	1	0	3.42e−04
MSCPSO vs ALPSO	0	2.85e+05	27	1	0	2.66e−04
MSCPSO vs MPBPSO	18.0901	1.84e+04	25	3	0	0.0352
MSCPSO vs DEPSO	11.0460	2.54e+04	26	2	0	0.0279
MSCPSO vs BiPop-CMA-ES	35.9279	2.10e+03	21	5	2	0.3630

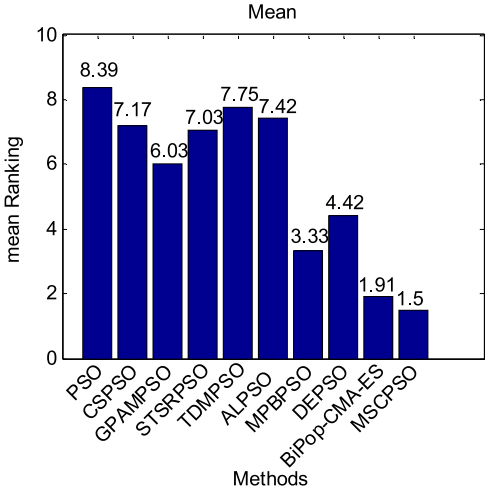

Fig. 9. Mean ranking of all CEC2013 functions of different algorithms in terms of Mean metric.

4.5. Experimental comparison and analysis on 28 CEC2013 benchmark functions

To further verify the performance of the proposed algorithm in different optimization problems, in this case, we used 28 benchmark functions of CEC2013 [40] which include 5 unimodal functions, 15 basic multimodal functions and 8 composition functions. We set uniformly the maximum number of function evaluations to be 300 000 and the dimension of all benchmark functions to be 30. Each algorithm was run independently over 30 times for all benchmark functions and then we calculated their statistical results concerning the final fitness values including maximum, minimum, mean, and standard deviation, which are shown in Table 6 (provided in Appendix B). The bold values denote the best results for each optimization function among all compared PSO algorithms.

As has been shown in Table 6, we can find that the proposed PSO algorithm performs best on nearly all optimization functions except Rotated Rosenbrock's Function and Composition Function 1 in terms of averaged fitness values. Although the proposed method cannot achieve best averaged fitness values on Rotated Rosenbrock's Function and Composition Function 1, their minimum fitness values are both global minimal optimum, which indicates that the proposed approach obtains the optimal values at least one time among all 30 runs on the two benchmark functions. In addition, from the results shown in Table 6, we

Table 5
Statistical results of different PSO algorithm on 5 basic benchmark functions.

Function	Algorithm	Max	Min	Mean	Std	Rank
Quadric	PSO	0.0031	5.6246e−06	4.2361e−04	7.7568e−04	10
	CSPSO	3.7046e−04	1.4493e−05	8.8128e−05	6.9776e−05	8
	GPAMPSO	1.2810e−12	3.0294e−15	1.9281e−13	2.6610e−13	3
	STSRPSO	6.7109e−05	3.1017e−08	5.2428e−06	1.2510e−05	4
	TDMPSO	4.9954e−04	8.7661e−08	5.2839e−05	1.1861e−04	7
	ALPSO	0.0025	7.3733e−07	1.8974e−04	5.0013e−04	9
	MPBPSO	6.4110e−13	6.5709e−15	1.3881e−13	1.6162e−13	2
	DEPSO	5.2647e−05	1.5414e−09	6.3765e−06	1.2778e−05	5
	BiPop-CMA-ES	7.0974e−13	0	5.3476e−14	1.0786e−14	1
	MSCPSO	3.3067e−04	0	2.4503e−05	1.0116e−04	6
Bent Cigar	PSO	1.6570e−08	9.2951e−19	7.2536e−10	3.0514e−09	7
	CSPSO	3.0705e−10	1.1237e−16	3.4899e−11	7.2861e−11	6
	GPAMPSO	1.8107e−18	1.2518e−24	1.4709e−19	3.7741e−19	1
	STSRPSO	0.9436	1.0483e−12	0.0394	0.1727	10
	TDMPSO	1.1819e−10	1.6464e−27	3.9479e−12	2.1578e−11	5
	ALPSO	3.8844e−07	2.2492e−17	1.3134e−08	7.0888e−08	9
	MPBPSO	1.3441e−17	6.2180e−23	1.0095e−18	3.0491e−18	2
	DEPSO	1.0695e−08	3.0912e−15	1.0208e−09	2.5550e−10	8
	BiPop-CMA-ES	2.3789e−15	0	1.6741e−16	9.0346e−17	4
	MSCPSO	3.1778e−16	0	1.0593e−17	5.8019e−17	3
Dminima	PSO	16.0216	8.4820	11.9377	2.3129	9
	CSPSO	16.9641	6.5971	10.8382	2.7864	4
	GPAMPSO	16.9641	7.5396	11.8748	2.4326	8
	STSRPSO	15.0792	6.5971	11.3722	2.2539	5
	TDMPSO	17.9065	5.6547	12.0005	2.7994	6
	ALPSO	17.9065	6.5971	11.3722	2.6649	10
	MPBPSO	1.4137e+01	7.5396e+00	1.0838e+01	2.0167e+00	3
	DEPSO	1.5079e+01	7.5396e+00	1.1498e+01	1.9957e+00	7
	BiPop-CMA-ES	4.5713e−10	4.5713e−10	4.5713e−10	0	1
	MSCPSO	4.5713e−10	4.5713e−10	4.5713e−10	0	2
r Griewank	PSO	1.6185e−01	0	2.5387e−02	4.0310e−02	8
	CSPSO	6.1083e−02	0	1.6707e−02	1.7612e−02	5
	GPAMPSO	2.4540e−01	0	2.7106e−02	5.4697e−02	10
	STSRPSO	5.8935e−02	9.4397e−12	1.8935e−02	1.7459e−02	6
	TDMPSO	7.8270e−02	0	1.3877e−02	1.9937e−02	3
	ALPSO	1.5936e−01	2.2204e−16	2.5785e−02	3.8446e−02	9
	MPBPSO	9.2808e−02	6.5836e−14	1.4110e−02	2.1414e−02	4
	DEPSO	1.2520e−01	0	2.2460e−02	3.1138e−02	7
	BiPop-CMA-ES	8.6744e−10	0	6.4308e−11	9.0379e−12	2
	MSCPSO	2.3429e−10	0	9.0830e−12	1.3102e−12	1
Schwefels	PSO	1.0248e+04	5.0147e+03	8.5597e+03	1.4229e+03	9
	CSPSO	7.1468e+03	4.2049e+03	5.7379e+03	793.5453	6
	GPAMPSO	9.3892e+03	3.5929e+03	6.5401e+03	1.4816e+03	7
	STSRPSO	6.9504e+03	3.7911e+03	5.6088e+03	729.2219	5
	TDMPSO	1.0038e+04	3.5335e+03	8.4142e+03	1.6387e+03	10
	ALPSO	1.0140e+04	4.9954e+03	8.5922e+03	1.2988e+03	8
	MPBPSO	6.4568e+03	3.1195e+03	5.3433e+03	7.8565e+02	4
	DEPSO	8.6845e+02	4.5801e+02	6.8689e+02	1.2976e+02	3
	BiPop-CMA-ES	4.9576e−04	4.0192e−04	4.3964e−04	6.3670e−06	2
	MSCPSO	3.8722e−04	3.8183e−04	3.8203e−04	9.8367e−07	1

can also find that the proposed approach can obtain the global minimal optimum for all runs in 4 unimodal functions and 10 basic multimodal functions. Compared to PSO variants, BiPop-CMA-ES still obtains more promising results on CEC test suite. Especially on Rotated Rosenbrock's Function and Composition Function 1, BiPop-CMA-ES achieves best averaged fitness values than our proposed PSO algorithm, which further shows its superiority in addressing single-objective optimization problems. Subsequently, the rank regarding mean value obtained by each compared algorithm in each of 28 optimization functions was also compared separately and the results are shown in Fig. 8. As shown in Fig. 8, our proposed algorithm performs well on most optimization problems and has the best performance compared to other algorithms.

In order to demonstrate the significant performance improvement of the proposed PSO algorithm over other PSO, all pairwise comparisons concerning the proposed PSO algorithm and other compared ones on 28 benchmark functions are shown in Table 4 which are obtained by wilcoxon signed ranks Test [24]. As has been shown in Table 4, the proposed PSO algorithm shows significant performance improvement over the other compared PSO variants with a level of significance $\alpha = 0.05$. The results further confirm that the proposed PSO algorithm outperforms other PSO ones with statistical significance. Compared to BiPop-CMA-ES, although the proposed PSO algorithm does not show significantly difference due to the obtained p-value bigger than a level of significance $\alpha = 0.05$, it can achieve at least comparative or even slightly better performance on most of the CEC test suite. Similar to the previous experiment, in order to intuitively compare the results, Fig. 9 shows the results of mean rankings for all compared variants on 28 benchmark functions in terms of Mean metric. As shown in Fig. 9, the proposed PSO algorithm ranks first followed by BiPop-CMA-ES and MPBPSO.

In order to further verify the effectiveness of the proposed PSO algorithm when dealing with more complicated optimization problems, we carried out the minimization optimization experiments on the 8 composition functions of CEC2013 with high dimensions. Similar to the previous cases, in order to avoid

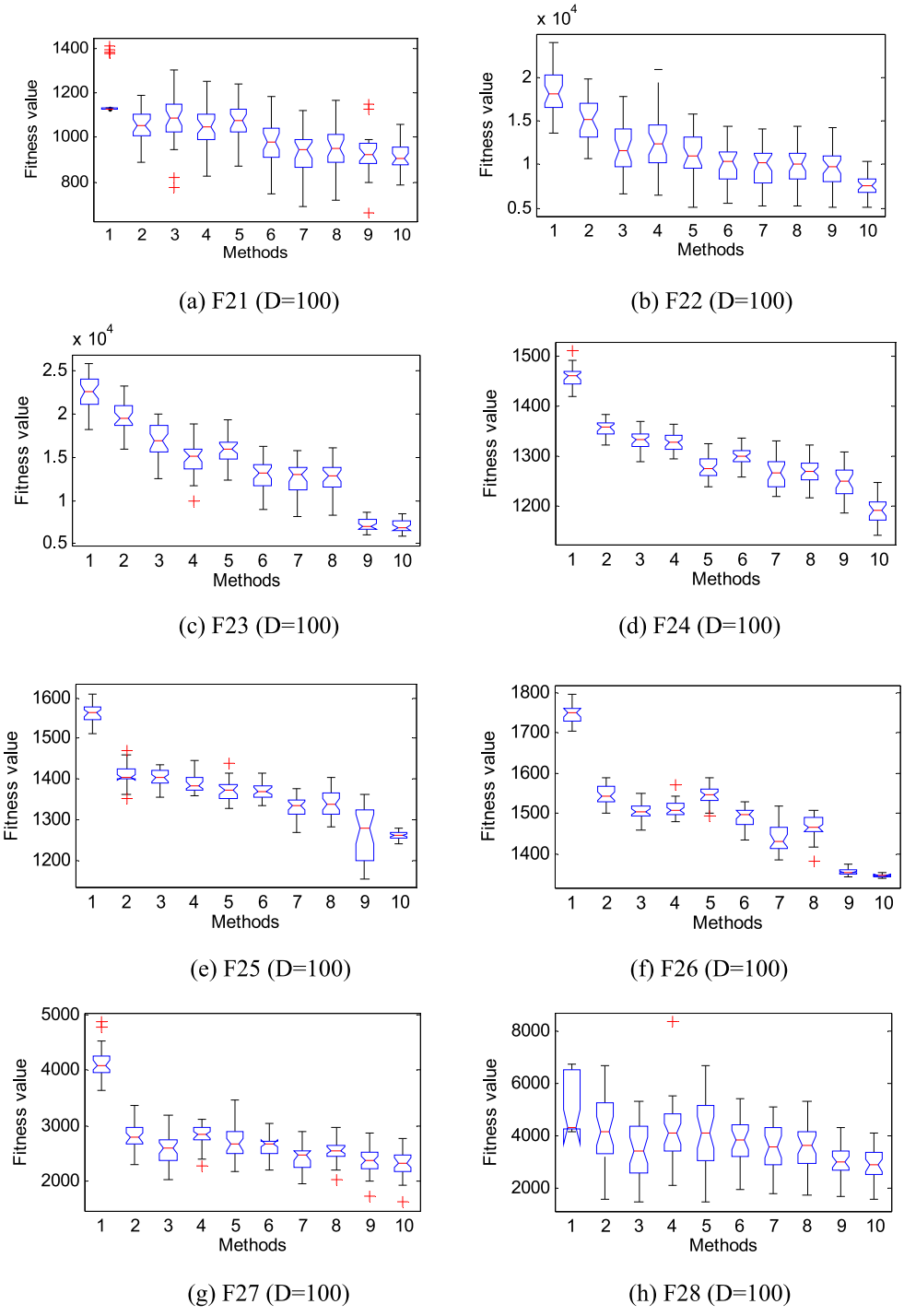

Fig. 10. Statistics comparisons on eight composition functions (D = 100) of different algorithms, where 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 denote PSO, CSPSO, GPAMPSO, STSRPSO, TDMPSO, ALPSO, MPBPSO, DEPSO, BiPop-CMA-ES, and MSCPSO, respectively.

the effect of randomness, all algorithms are independently run over 20 times and then their fitness values of the final solution obtained by each algorithm on all functions with 100 dimensions are statistically shown in Fig. 10.

From the results shown in Fig. 10, we can find that the proposed PSO algorithm shows the lowest mean fitness value on all composition functions with 100 dimensions. Specifically, for F21, F24, F25, and F26 benchmark composition functions with 100 dimensions, all compared PSO algorithms can also obtain relatively satisfactory results. The only difference is that other compared PSO algorithms fail to yield the comparable accuracy of the final solutions with the proposed PSO algorithm on F27 and F28 besides F22 and F23 functions due to the effect of high dimensions. The experimental results indicate that the proposed PSO algorithm outperforms other compared PSO ones when dealing with the more complicated optimization problems. Compared to BiPop-CMA-ES, the proposed PSO still achieves comparative results on the tested high-dimensional complex optimization problems. This primarily attributes to the application of multi-scale mutation strategy and appropriate self-adaptive mutation threshold setting, which can guarantee the population sufficiently explore the global solution space at the early stage and accurately

Table 6
Statistical results of different PSO algorithm on 28 benchmark functions (D = 30).

Function	Algorithm	Max	Min	Mean	Std	Rank
Sphere	PSO	3.1403e−15	3.0611e−23	4.5142e−17	1.2134e−17	7
	CSPSO	2.0014e−15	4.1097e−23	5.9401e−16	3.5417e−16	9
	GPAMPSO	6.3147e−24	1.6746e−28	7.0937e−25	1.2045e−24	3
	STSRPSO	5.1403e−07	1.9750e−16	4.3316e−08	2.3471e−07	10
	TDMPSO	3.4197e−18	1.5607e−28	6.1120e−19	7.6454e−18	6
	ALPSO	6.1471e−15	5.2217e−22	4.6174e−16	3.2671e−15	8
	MPBPSO	4.1507e−22	0	3.9094e−23	1.3274e−23	4
	DEPSO	2.3147e−23	0	6.1964e−23	5.7414e−23	5
	BiPop-CMA-ES	0	0	0	0	1
	MSCPSO	0	0	0	0	1
Rotated High Conditioned Elliptic	PSO	7.3149e+05	9.1340e+04	3.1147e+05	2.1046e+05	10
	CSPSO	1.6346e+06	7.3586e+04	2.9545e+05	3.1996e+05	8
	GPAMPSO	2.0846e+05	4.2216e+04	2.3471e+05	6.0127e+04	5
	STSRPSO	7.2013e+05	4.9307e+04	3.0143e+05	1.9206e+05	4
	TDMPSO	5.5617e+05	3.6101e+04	2.4687e+05	1.9631e+05	6
	ALPSO	7.1283e+05	8.3346e+04	2.7754e+05	1.3676e+05	7
	MPBPSO	1.5536e+03	6.3146e+02	1.5667e+03	2.9103e+02	3
	DEPSO	6.3794e+04	4.6698e+03	3.0069e+04	7.3664e+03	9
	BiPop-CMA-ES	1.8517e+03	0	7.2478e+02	6.0477e+01	2
	MSCPSO	2.5048e+03	0	5.4813e+02	3.3318e+01	1
Rotated Bent Cigar	PSO	5.3916e+04	2.3319e−02	5.6794e+03	4.3167e+03	10
	CSPSO	8.4517e+02	6.5594e−01	2.0367e+02	3.6109e+02	5
	GPAMPSO	1.0397e+04	3.5267e−04	8.3376e+02	3.6108e+03	6
	STSRPSO	9.1167e+03	1.0354e−02	2.3607e+03	2.1030e+03	9
	TDMPSO	4.5127e+02	1.2531e−02	2.0135e+02	1.3607e+02	4
	ALPSO	5.0377e+03	2.1413e−02	8.6335e+02	1.0773e+03	7
	MPBPSO	4.6339e+02	0	1.2236e+02	5.3546e+00	3
	DEPSO	5.3479e+03	0	9.7894e+02	1.0128e+01	8
	BiPop-CMA-ES	6.2043e+01	0	9.0174e+00	4.3178e−01	2
	MSCPSO	4.5596e+01	0	2.3365e+00	5.0547e−01	1
Rotated Discus	PSO	3.6735e+03	1.9354e+02	7.6634e+02	4.6531e+02	10
	CSPSO	1.4538e+03	8.5537e+01	6.5397e+02	4.0179e+02	8
	GPAMPSO	7.5478e+02	3.6641e+01	3.6749e+02	1.9734e+02	7
	STSRPSO	9.0234e+02	9.6647+00	2.0491e+02	3.0471e+02	5
	TDMPSO	7.6355e+02	1.1964e+02	3.6745e+02	2.6635e+02	6
	ALPSO	2.0386e+03	7.6634e+01	6.9913e+02	5.6784e+02	9
	MPBPSO	4.5673e+02	5.6675e+00	3.0416e+01	5.6794e+00	3
	DEPSO	6.3374e+02	9.6355e+00	1.0567e+02	3.0069e+01	4
	BiPop-CMA-ES	1.3047e+01	1.1903e+00	4.1473e+00	9.1340e−01	2
	MSCPSO	6.6049e+00	1.4571e+00	3.0567e+00	4.6274e−01	1
Different Powers	PSO	1.0297e−11	2.5697e−18	6.4788e−13	3.6478e−12	8
	CSPSO	4.5678e−08	9.5467e−11	9.1067e−09	2.3475e−08	9
	GPAMPSO	8.3497e−21	6.3435e−25	3.6673e−22	5.0697e−22	4
	STSRPSO	3.5679e−05	5.2201e−11	6.3447e−06	2.0137e−06	10
	TDMPSO	6.0323e−14	5.3415e−22	9.5879e−15	3.3146e−14	6
	ALPSO	5.3420e−12	9.3006e−18	2.5673e−13	7.1255e−13	7
	MPBPSO	5.3026e−16	3.6674e−26	1.3348e−22	4.1294e−23	3
	DEPSO	1.0199e−18	7.1496e−25	3.4675e−20	3.1583e−22	5
	BiPop-CMA-ES	0	0	0	0	1
	MSCPSO	0	0	0	0	1
Rotated Rosenbrock	PSO	3.1045e−08	9.6375e−15	5.6673e−10	5.2049e−10	6
	CSPSO	2.3679e−12	3.0146e−15	6.3577e−14	3.1106e−14	4
	GPAMPSO	8.0067e−21	5.6741e−23	2.3066e−21	1.3576e−21	2
	STSRPSO	6.5597e−07	1.0378e−11	6.0378e−08	1.9367e−07	9
	TDMPSO	2.0374e−06	3.6009e−14	1.3479e−07	1.0037e−07	10
	ALPSO	1.3476e−07	5.2214e−13	3.0146e−09	6.3765e−09	8
	MPBPSO	6.3471e−13	3.0144e−16	1.3845e−14	3.5421e−14	3
	DEPSO	5.2143e−08	4.5127e−12	1.3751e−09	1.9871e−10	7
	BiPop-CMA-ES	5.3146e−20	0	4.2374e−22	3.3761e−13	1
	MSCPSO	1.0445e−08	0	1.1413e−10	5.1023e−11	5
Rotated Schaffers F7	PSO	2.3479e+00	6.3117e−03	7.6346e−01	3.1445e−01	10
	CSPSO	3.4545e+00	3.3374e−03	5.2147e−01	2.1451e−01	7
	GPAMPSO	2.5457e+00	9.4558e−04	6.4751e−01	6.1744e−01	9
	STSRPSO	6.1572e−01	5.1482e−03	1.1210e−01	3.1247e−01	6
	TDMPSO	3.1424e−01	2.6973e−04	7.8493e−02	4.4513e−02	5
	ALPSO	5.7496e−01	6.4214e−03	5.2177e−01	1.4727e−01	8
	MPBPSO	3.5514e−01	9.8423e−05	6.4714e−02	5.5749e−02	4
	DEPSO	6.4531e−01	1.3741e−05	3.4417e−03	2.3741e−03	3
	BiPop-CMA-ES	1.3140e−01	0	2.1463e−03	1.4794e−04	2
	MSCPSO	2.1434e−02	0	6.1545e−04	5.4521e−05	1

(*continued on next page*)

Table 6 (*continued*).

Function	Algorithm	Max	Min	Mean	Std	Rank
Rotated Ackley	PSO	2.3117e+01	2.1842e+01	2.1539e+01	5.5074e−02	7
	CSPSO	2.2120e+01	2.1873e+01	2.1550e+01	5.2080e−02	10
	GPAMPSO	2.1108e+01	2.3797e+01	2.1506e+01	6.2740e−02	5
	STSRPSO	2.3075e+01	2.2803e+01	2.1540e+01	5.0557e−02	8
	TDMPSO	2.1117e+01	2.1927e+01	2.1549e+01	5.7888e−02	9
	ALPSO	2.3060e+01	2.1841e+01	2.1534e+01	4.6616e−02	6
	MPBPSO	1.1429e+01	5.6822e−01	1.0067e+00	8.3980e−02	4
	DEPSO	8.1301e+00	2.0459e−01	5.1678e−01	6.2605e−02	2
	BiPop-CMA-ES	9.3476e+00	3.1762e−02	6.3122e−01	3.2067e−02	3
	MSCPSO	4.8958e−01	9.3259e−14	3.8462e−03	1.0874e−02	1
Rotated Weierstrass	PSO	3.9217e+01	1.8308e+01	2.5627e+01	5.2082e+00	8
	CSPSO	3.0064e+01	2.2933e+01	2.9417e+01	3.9885e+00	10
	GPAMPSO	3.1712e+01	1.3875e+01	2.6724e+01	6.5947e+00	9
	STSRPSO	3.6277e+01	1.0377e+01	2.4684e+01	7.2652e+00	5
	TDMPSO	3.4103e+01	1.6963e+01	2.5375e+01	6.7257e+00	7
	ALPSO	3.7170e+01	1.5039e+01	2.5013e+01	4.6471e+00	6
	MPBPSO	1.0049e+01	5.6187e−01	8.7253e−01	1.5567e−01	4
	DEPSO	5.9551e+00	3.5588e−01	5.4707e−01	3.1451e−02	3
	BiPop-CMA-ES	1.3407e+00	0	9.4761e−02	1.0234e−02	2
	MSCPSO	3.4856e−01	0	2.0546e−02	7.2975e−03	1
Rotated Griewank	PSO	6.6761e+01	2.5073e+01	3.8020e+01	1.0494e+01	9
	CSPSO	5.8006e+01	2.5073e+01	3.7662e+01	9.0464e+00	8
	GPAMPSO	7.4423e+01	2.0894e+01	4.1710e+01	1.3585e+01	10
	STSRPSO	4.6761e+01	2.9252e+01	2.2376e+01	8.7196e+00	5
	TDMPSO	6.6461e+01	2.1939e+01	3.1248e+01	1.4103e+01	7
	ALPSO	5.9895e+01	1.8804e+01	2.3657e+01	1.2369e+01	6
	MPBPSO	9.4164e−01	7.6820e−02	3.6533e−01	6.2087e−02	4
	DEPSO	1.3726e−01	5.5804e−04	1.6828e−02	1.2369e−03	3
	BiPop-CMA-ES	4.3726e−02	1.3476e−07	5.0609e−03	2.3147e−03	2
	MSCPSO	6.6489e−02	0	8.9958e−05	2.0708e−05	1
Rastrigin	PSO	6.3471e+01	1.8804e+01	4.3965e+01	1.6416e+01	10
	CSPSO	6.5667e+01	2.5073e+01	3.7712e+01	9.1615e+00	7
	GPAMPSO	5.2234e+01	1.5073e+01	3.4425e+01	1.1851e+01	6
	STSRPSO	4.4572e+01	2.2983e+01	3.0327e+01	9.9708e+00	5
	TDMPSO	7.6612e+01	2.4284e+01	4.3700e+01	1.2441e+01	8
	ALPSO	7.6612e+01	2.2983e+01	4.3810e+01	1.1705e+01	9
	MPBPSO	9.3723e+00	4.5482e+00	6.6726e+00	2.6123e−01	4
	DEPSO	6.0306e+00	1.6209e+00	3.3990e+00	6.9333e−01	3
	BiPop-CMA-ES	3.1453e+00	9.6341e−05	4.4730e−01	1.5354e−01	2
	MSCPSO	1.7897e+00	0	5.2054e−02	1.2049e−02	1
Rotated Rastrigin	PSO	1.3778e+02	3.5667e+01	6.8357e+01	2.3463e+01	9
	CSPSO	1.2695e+02	3.1341e+01	6.5435e+01	2.3983e+01	5
	GPAMPSO	1.8773e+02	3.2393e+01	6.6278e+01	3.2367e+01	6
	STSRPSO	1.6089e+02	2.9258e+01	6.8107e+01	3.0786e+01	7
	TDMPSO	1.1135e+02	3.3488e+01	6.8271e+01	2.1016e+01	8
	ALPSO	1.3646e+02	4.6937e+01	7.5744e+01	2.3357e+01	10
	MPBPSO	3.5745e+01	4.2019e+00	9.8274e+00	4.6028e−01	4
	DEPSO	1.1164e+01	1.1164e+00	7.6395e+00	2.5387e−01	3
	BiPop-CMA-ES	6.3471e+00	0	2.3167e−01	1.5387e−01	2
	MSCPSO	2.1327e+00	0	8.2072e−02	2.2561e−02	1
Non-Continuous Rotated Rastrigin	PSO	2.3986e+02	8.5657e+01	1.3904e+02	3.6478e+01	9
	CSPSO	3.3794e+02	8.1228e+01	1.3076e+02	3.5040e+01	8
	GPAMPSO	1.7736e+02	8.0127e+01	1.0029e+02	2.5333e+01	5
	STSRPSO	2.0727e+02	5.3837e+01	1.1810e+02	3.7538e+01	6
	TDMPSO	1.8439e+02	4.1721e+01	1.2332e+02	2.8624e+01	7
	ALPSO	2.4256e+02	8.0248e+01	1.4280e+02	4.1717e+01	10
	MPBPSO	1.9092e+02	3.0092e+01	8.7375e+01	4.5630e+00	4
	DEPSO	9.1190e+01	1.2650e+01	4.9343e+01	1.3225e+00	3
	BiPop-CMA-ES	2.5761e+01	0	8.2253e−01	7.3143e−02	2
	MSCPSO	8.0286e+00	0	1.4180e−01	3.2182e−02	1
Schwefel	PSO	5.5187e+03	1.6323e+03	3.5798e+03	8.9053e+02	6
	CSPSO	4.6812e+03	2.6101e+03	3.5373e+03	6.9680e+02	5
	GPAMPSO	4.9871e+03	2.4058e+03	3.4529e+03	7.3136e+02	4
	STSRPSO	5.1839e+03	2.4452e+03	3.5919e+03	7.7350e+02	8
	TDMPSO	4.7447e+03	2.4239e+03	3.8652e+03	6.8288e+02	10
	ALPSO	5.2142e+03	1.9494e+03	3.6662e+03	9.1542e+02	9
	MPBPSO	2.8082e+03	5.5501e+02	8.3372e+02	4.9507e+01	3
	DEPSO	6.4917e+03	1.6311e+03	3.5855e+03	5.5922e+02	7
	BiPop-CMA-ES	1.0276e+03	4.3175e+02	8.1745e+02	3.1344e+01	2
	MSCPSO	8.4086e+02	7.0092e+01	1.1745e+02	1.1016e+01	1

(*continued on next page*)

Table 6 (*continued*).

Function	Algorithm	Max	Min	Mean	Std	Rank
Rotated Schwefel	PSO	5.2256e+03	2.7761e+03	3.8000e+03	6.1789e+02	5
	CSPSO	5.4082e+03	2.6074e+03	4.0482e+03	7.0928e+02	7
	GPAMPSO	4.5461e+03	2.3645e+03	3.8655e+03	3.8674e+02	6
	STSRPSO	5.6829e+03	2.3964e+03	4.2433e+03	6.6527e+02	9
	TDMPSO	5.5850e+03	3.0382e+03	4.1211e+03	9.4396e+02	8
	ALPSO	5.4245e+03	2.6592e+03	3.7542e+03	6.7076e+02	4
	MPBPSO	3.1895e+03	6.4406e+02	9.5490e+02	2.3442e+01	3
	DEPSO	6.6628e+03	1.3712e+03	4.5409e+03	3.6549e+02	10
	BiPop-CMA-ES	1.5376e+03	5.3021e+00	8.0337e+02	3.0397e+01	2
	MSCPSO	8.3625e+02	1.0092e+01	3.3599e+02	1.2975e+01	1
Rotated Katsuura	PSO	3.9599e+01	5.0254e+00	1.4689e+01	8.0076e+00	10
	CSPSO	2.8852e+01	2.6515e+00	1.1442e+01	7.6924e+00	7
	GPAMPSO	2.0137e+01	3.0507e+00	1.1727e+01	5.2974e+00	8
	STSRPSO	2.1362e+01	7.6839e−01	5.6109e+00	4.5805e+00	4
	TDMPSO	2.1828e+01	3.1633e+00	1.1376e+01	5.9153e+00	6
	ALPSO	3.7733e+01	2.9149e+00	1.3433e+01	9.1125e+00	9
	MPBPSO	8.3174e+00	3.4710e−02	1.4473e+00	7.3614e−01	3
	DEPSO	1.5601e+01	9.4413e−01	6.3471e+00	1.8064e+00	5
	BiPop-CMA-ES	5.3174e+00	1.0376e−11	5.3146e−01	1.3467e−01	2
	MSCPSO	3.2555e+00	3.8369e−14	2.6837e−01	1.9242e−01	1
Lunacek Bi_Rastrigin	PSO	2.5331e+02	6.7072e+01	1.0178e+02	3.9707e+01	9
	CSPSO	1.8320e+02	5.0831e+01	9.2900e+01	3.6780e+01	7
	GPAMPSO	1.6169e+02	3.3724e+01	8.8355e+01	2.9135e+01	5
	STSRPSO	5.2117e+02	3.8270e+01	1.0097e+02	9.7152e+01	8
	TDMPSO	2.3087e+02	5.8529e+01	1.0759e+02	3.8759e+01	10
	ALPSO	1.6164e+02	5.1841e+01	9.2090e+01	2.4159e+01	6
	MPBPSO	9.7856e+01	1.1503e+01	3.3482e+01	4.5614e+00	3
	DEPSO	1.6322e+02	1.4144e+01	6.9409e+01	2.9135e+00	4
	BiPop-CMA-ES	6.6347e+00	0	8.0314e−03	1.3047e−03	1
	MSCPSO	8.1089e+00	0	5.5962e−02	5.3365e−03	2
Rotated Lunacek Bi_Rastrigin	PSO	2.2226e+02	9.9501e+01	1.5041e+02	3.4396e+01	8
	CSPSO	1.9306e+02	6.9897e+01	1.0747e+02	3.2676e+01	7
	GPAMPSO	1.5085e+02	5.9889e+01	9.6629e+01	2.4158e+01	6
	STSRPSO	1.4784e+02	6.5397e+01	9.4396e+01	2.0932e+01	5
	TDMPSO	2.0925e+02	1.0793e+02	1.5998e+02	2.5432e+01	10
	ALPSO	2.1613e+02	1.0515e+02	1.5249e+02	3.2660e+01	9
	MPBPSO	5.4760e+01	1.6476e+01	3.8583e+01	8.4221e+00	3
	DEPSO	1.3996e+02	5.8231e+01	8.8159e+01	9.3173e+00	4
	BiPop-CMA-ES	6.2678e+00	0	3.4567e−01	1.5367e−01	2
	MSCPSO	4.2678e+00	0	1.3720e−01	6.9531e−02	1
Expanded Griewank's plus Rosenbrock	PSO	1.6803e+01	6.1287e+00	1.0265e+01	2.9903e+00	9
	CSPSO	1.4436e+02	1.1341e+01	5.7777e+01	3.4831e+01	10
	GPAMPSO	1.5791e+01	6.6005e+00	9.6755e+00	2.4280e+00	6
	STSRPSO	1.4644e+01	5.2026e+00	1.0009e+01	2.3922e+00	8
	TDMPSO	1.5927e+01	3.6470e+00	8.8264e+00	3.0924e+00	5
	ALPSO	1.7817e+01	3.8853e+00	9.7478e+00	3.4085e+00	7
	MPBPSO	3.6570e+00	5.7447e−01	1.0652e+00	1.5715e−01	2
	DEPSO	9.7431e+00	2.1384e+00	7.4501e+00	7.9690e−01	4
	BiPop-CMA-ES	3.6570e+00	0	1.0652e+00	1.5715e−01	2
	MSCPSO	1.1710e+00	0	7.4145e−01	1.9357e−01	1
Expanded Scaffer's F6	PSO	1.5370e+01	9.8337e+00	1.2857e+01	1.8816e+00	8
	CSPSO	1.5381e+01	8.9312e+00	1.2925e+01	1.9473e+00	9
	GPAMPSO	1.5370e+01	7.7996e+00	1.2376e+01	1.9373e+00	5
	STSRPSO	1.5381e+01	9.1134e+00	1.2796e+01	1.9086e+00	7
	TDMPSO	1.5900e+01	6.9440e+00	1.3574e+01	1.8619e+00	10
	ALPSO	1.5515e+01	7.4631e+00	1.2414e+01	2.0066e+00	6
	MPBPSO	1.4265e+01	4.9275e+00	8.8621e+00	8.2762e−01	4
	DEPSO	1.5031e+01	3.1966e+00	6.3179e+00	9.3141e−01	3
	BiPop-CMA-ES	1.3541e+01	2.3477e+00	4.2617e+00	5.3679e−01	2
	MSCPSO	1.2419e+01	1.6901e+00	3.9641e+00	4.5519e−01	1
Composition function 1	PSO	1.0957e+02	1.0554e+02	1.0470e+02	9.9808e−01	9
	CSPSO	1.0602e+02	1.0501e+02	1.0301e+02	2.9313e−03	5
	GPAMPSO	1.0601e+02	1.0240e+02	1.0300e+02	1.7235e−03	4
	STSRPSO	1.0612e+02	1.0502e+02	1.0305e+02	2.5572e−02	6
	TDMPSO	1.0972e+02	1.0477e+02	1.0456e+02	9.4427e−01	8
	ALPSO	1.1360e+02	1.0493e+02	1.0486e+02	1.5267e+00	10
	MPBPSO	1.0833e+02	1.0460e+02	1.0264e+02	8.2964e−01	2
	DEPSO	1.0840e+02	1.0463e+02	1.0275e+02	2.3947e+00	3
	BiPop-CMA-ES	1.0535e+02	1.0240e+02	1.0194e+02	1.9947e+00	1
	MSCPSO	1.0958e+02	1.0240e+02	1.0358e+02	1.9831e−01	7

(*continued on next page*)

Table 6 (*continued*).

Function	Algorithm	Max	Min	Mean	Std	Rank
Composition function 2	PSO	5.0984e+03	2.8063e+03	3.9518e+03	6.3922e+02	10
	CSPSO	4.6594e+03	3.0317e+03	3.6735e+03	4.9071e+02	7
	GPAMPSO	5.1203e+03	2.0965e+03	3.6461e+03	7.3455e+02	6
	STSRPSO	4.8914e+03	2.8774e+03	3.8269e+03	4.9332e+02	9
	TDMPSO	4.5511e+03	2.7084e+03	3.5640e+03	4.6873e+02	5
	ALPSO	5.6125e+03	2.9325e+03	3.7950e+03	6.4013e+02	8
	MPBPSO	2.0424e+03	1.4064e+02	4.7812e+02	5.5109e+01	3
	DEPSO	9.0109e+02	3.7418e+02	6.0255e+02	1.4278e+01	4
	BiPop-CMA-ES	5.4214e+02	1.1319e+02	3.0047e+02	7.6341e+01	2
	MSCPSO	2.7016e+02	1.1882e+02	1.9375e+02	1.0167e+00	1
Composition function 3	PSO	6.4441e+03	2.5684e+03	4.1439e+03	1.0122e+03	5
	CSPSO	5.6983e+03	2.8687e+03	4.2022e+03	7.9678e+02	6
	GPAMPSO	6.2388e+03	2.7369e+03	4.3553e+03	7.8908e+02	8
	STSRPSO	5.7881e+03	3.2502e+03	4.5327e+03	5.9144e+02	10
	TDMPSO	6.8019e+03	2.7992e+03	4.4584e+03	8.5248e+02	9
	ALPSO	8.4294e+03	3.1605e+03	4.3361e+03	1.1208e+03	7
	MPBPSO	1.6100e+03	4.5019e+02	8.8263e+02	6.1657e+01	4
	DEPSO	1.7901e+03	3.6046e+02	8.5774e+02	1.0682e+01	3
	BiPop-CMA-ES	5.6746e+02	1.9674e+02	2.6683e+02	5.1673e+01	2
	MSCPSO	1.5941e+02	1.1882e+02	1.2253e+02	8.0252e+00	1
Composition function 4	PSO	3.6695e+02	1.8772e+02	2.3849e+02	4.1957e+01	7
	CSPSO	3.2344e+02	1.9656e+02	2.3096e+02	3.7299e+01	5
	GPAMPSO	3.2284e+02	1.9093e+02	2.3882e+02	3.7406e+01	8
	STSRPSO	2.9943e+02	2.0459e+02	2.3278e+02	2.5368e+01	6
	TDMPSO	3.4500e+02	1.9702e+02	2.4953e+02	3.3898e+01	9
	ALPSO	3.3754e+02	1.9201e+02	2.3001e+02	3.3930e+01	4
	MPBPSO	3.1008e+02	1.5408e+02	1.6222e+02	5.6873e+00	2
	DEPSO	3.2439e+02	2.0610e+02	2.6034e+02	9.7782e+00	10
	BiPop-CMA-ES	1.8567e+02	1.5374e+02	1.6748e+02	4.6779e+00	3
	MSCPSO	1.3726e+02	1.1883e+02	1.1921e+02	3.8966e+00	1
Composition function 5	PSO	3.2767e+02	1.6892e+02	2.3893e+02	4.4058e+01	9
	CSPSO	3.2634e+02	1.8275e+02	2.2868e+02	4.2749e+01	6
	GPAMPSO	3.0891e+02	1.8901e+02	2.2890e+02	3.6977e+01	7
	STSRPSO	2.6590e+02	1.8582e+02	2.1664e+02	2.2538e+01	5
	TDMPSO	3.5790e+02	1.8240e+02	2.3998e+02	3.9923e+01	10
	ALPSO	3.3313e+02	1.7730e+02	2.3628e+02	4.2691e+01	8
	MPBPSO	2.2058e+02	1.3725e+02	1.7380e+02	8.1920e+00	4
	DEPSO	2.0944e+02	1.3068e+02	1.5947e+02	6.2085e+00	3
	BiPop-CMA-ES	1.7534e+02	1.1479e+02	1.3067e+02	3.0469e+00	1
	MSCPSO	1.8954e+02	1.1883e+02	1.4056e+02	2.6892e+00	2
Composition function 6	PSO	5.1712e+02	3.2229e+02	3.9568e+02	4.5043e+01	8
	CSPSO	5.5621e+02	3.6048e+02	4.1523e+02	4.7164e+01	10
	GPAMPSO	4.7801e+02	2.6499e+02	3.7954e+02	4.9556e+01	6
	STSRPSO	4.4307e+02	3.3092e+02	3.6077e+02	2.4361e+01	4
	TDMPSO	5.3582e+02	3.1352e+02	4.0000e+02	5.3034e+01	9
	ALPSO	5.7011e+02	2.1165e+02	3.9412e+02	7.4002e+01	7
	MPBPSO	4.6731e+02	3.3472e+02	3.6527e+02	1.5326e+01	5
	DEPSO	3.9451e+02	2.5193e+02	3.5183e+02	6.3712e+00	3
	BiPop-CMA-ES	4.0195e+02	2.3167e+02	3.2691e+02	5.5633e+00	2
	MSCPSO	3.2427e+02	1.9592e+02	1.9608e+02	5.5633e+00	1
Composition function 7	PSO	6.4871e+03	2.2907e+03	3.7436e+03	1.0514e+03	10
	CSPSO	4.5716e+03	1.9651e+03	3.2740e+03	6.2784e+02	5
	GPAMPSO	5.6308e+03	2.3892e+03	3.5480e+03	6.9609e+02	7
	STSRPSO	4.6244e+03	2.3659e+03	3.2903e+03	4.8197e+02	6
	TDMPSO	5.7994e+03	2.8153e+03	3.7433e+03	7.1386e+02	9
	ALPSO	4.9516e+03	2.6086e+03	3.5654e+03	6.2857e+02	8
	MPBPSO	7.6889e+02	4.7460e+02	6.1820e+02	1.3894e+01	3
	DEPSO	1.1427e+03	5.8604e+02	8.8698e+02	4.0585e+01	4
	BiPop-CMA-ES	2.0167e+02	1.7431e+02	1.9873e+02	5.0164e+00	1
	MSCPSO	3.1599e+02	1.9591e+02	2.2308e+02	8.1890e+00	2
Composition function 8	PSO	4.3673e+03	1.0007e+03	1.9358e+03	7.5309e+02	9
	CSPSO	3.2142e+03	1.1460e+03	1.8479e+03	5.3612e+02	7
	GPAMPSO	2.7396e+03	1.0266e+03	1.7880e+03	4.4220e+02	6
	STSRPSO	2.6457e+03	1.2428e+03	1.8645e+03	3.8624e+02	8
	TDMPSO	3.2410e+03	1.1617e+03	2.0109e+03	4.6364e+02	10
	ALPSO	3.3039e+03	1.1675e+03	1.7238e+03	4.3418e+02	5
	MPBPSO	6.9242e+02	2.0581e+02	4.1811e+02	6.6836e+01	2
	DEPSO	1.5954e+03	5.9133e+02	9.4041e+02	6.1078e+01	4
	BiPop-CMA-ES	7.6147e+02	3.3471e+02	5.6149e+02	7.9167e+01	3
	MSCPSO	3.7765e+02	1.0442e+02	2.4339e+02	5.0595e+01	1

exploit local solution space at the later stage. The well-balance between exploration and exploitation can help the proposed PSO algorithm effectively avoid the premature convergence and thus improve the accuracy of the final solution.

5. Conclusion

In the paper, we proposed a Multiple Scale self-adaptive cooperative mutation strategy-based particle swarm optimization. In the proposed PSO algorithm, multi-scale mutation strategy is firstly introduced to guarantee the populations to sufficiently search in the whole solution space during evolution. On one hand, large-scale mutation can help the populations explore the global solution space and rapidly locate the better solution area at the early stage. On the other hand, small-scale mutation can make the populations more accurately exploit the local best solution area, eventually enlarging the possibility of finding the better solution. In addition, we also adopt the self-adaptive mutation threshold setting method for each optimization variable, which can not only make the populations have more chance to participate in mutation at the early stage, but also guarantee the PSO inherent evolution to exploit the more accurate solution space during the later evolutionary stage. Numerical experimental results on basic benchmark functions and CEC'2013 benchmark functions with the dimensional of 30 and 100 demonstrated that compared to other PSO algorithms, the proposed PSO algorithm can more remarkably improve the accuracy of the final solution and convergence speed when dealing with different dimensional optimization problems. It is worth noting that although some advances have been made in some cases, the proposed PSO algorithm cannot obtain satisfactory accuracy of the final solution especially on the composition functions with rotate. Good results would be expected in the proposed algorithm if the population structure and mechanism of information exchanging among individuals are further improved, which is also the focus of our future work.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2020.106124.

CRediT authorship contribution statement

Xinmin Tao: Conceptualization, Methodology, Software, Writing - original draft. **Wenjie Guo:** Software, Investigation, Formal analysis. **Qing Li:** Data curation, Investigation. **Chao Ren:** Writing - review & editing. **Rui Liu:** Visualization.

Acknowledgments

This work was supported in part by the Fundamental Research Funds for the Central Universities, China no. 2572017EB02, 2572017CB07, Innovative talent fund of Harbin science and technology Bureau, China (No. 2017RAXXJ018), Double first-class scientific research foundation of Northeast Forestry University, China (411112438). We thank Qing He and Junrong Zhou for their assistance for the experiment designs in the revised manuscript. The authors would like to thanks anonymous reviewers for their constructive comments.

Appendix A

See Table 5.

Appendix B

See Table 6.

References

[1] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[2] H.G. Beyer, H.P. Schwefel, Evolution strategies-A comprehensive introduction, Nat. Comput. 1 (1) (2002) 3–52.
[3] G. Lin, J. Guan, A hybrid binary particle swarm optimization for the obnoxious p-median problem, Inform. Sci. 425 (2017) 1–17.
[4] J. Han, G. Zhang, Y. Hu, J. Lu, A solution to bi/tri-level programming problems using particle swarm optimization, Inform. Sci. 370 (2016) 519–537.
[5] J.O. Pedro, D. Muhammed, O.A. Dahunsi, A.M. Montaz, Dynamic neural network-based feedback linearization control of full-car suspensionsusing pso, Appl. Soft. Comput. 70 (2018) 723–736.
[6] H. Pang, F. Liu, Z. Xu, Variable universe fuzzy control for vehicle semi-active suspension system with MR damper combining fuzzy neural network and particle swarm optimization, Neurocomputing 306 (2018) 130–140.
[7] N. Kalaiarasi, S. Paramasivam, S.S. Dash, P. Sanjeevikumar, L. Mihet-Popa, PSO based MPPT implementation in dspace controller integrated through z-source inverter for photovoltaic applications, Energies 9 (2017) 1–10.
[8] Q. Zhang, W. Liu, X. Meng, et al., Vector coevolving particle swarm optimization algorithm, Inform. Sci. 394 (2017) 273–298.
[9] Chaudhary Divya, K. Bijendra, Cloudy GSA for load scheduling in cloud computing, Appl. Soft. Comput. 71 (2018) 861–871.
[10] J. Li, Y. Zhang, X. Chen, Y. Xiang, Secure attribute-based data sharing for resource-limited users in cloud computing, Comput. Secur. 72 (2018) 1–12.
[11] P. Agarwalla, S. Mukhopadhyay, Efficient player selection strategy based diversified particle swarm optimization algorithm for global optimization, Inform. Sci. 397 (2017) 69–90.
[12] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings of the IEEE Congress on Evolutionary Computation, 1998, pp. 69–73.
[13] Y. Shi, R.C. Eberhart, Particle swarm optimization with fuzzy adaptive inertia weight, in: Proceedings of Workshop Particle Swarm Optimization, Indianapolis, IN, 2001, pp. 101–106.
[14] N.J. Li, W.J. Wang, C.C.J. Hsu, Hybrid particle swarm optimization incorporating fuzzy reasoning and weighted particle, Neurocomputing 167 (2015) 488–501.
[15] J.J.D. Nesamalar, P. Venkatesh, S.C. Raja, Managing multi-line power congestion by using hybrid Nelder–Mead – fuzzy adaptive particle swarm optimization (HNM-FAPSO), Appl. Soft Comput. 43 (2016) 222–234.
[16] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, Q. Tian, Self-adaptive learning based particle swarm optimization, Inform. Sci. 191 (2011) 4515–4538.
[17] K. Chen, F. Zhou, L. Yin, et al., A hybrid particle swarm optimizer with sine cosine acceleration coefficients, Inform. Sci. 422 (2017) 218–241.
[18] Z. Michalewicz, A hierarchy of evolution programs, in: Genetic Algorithms + Data Structures = Evolution Programs, 1996, pp. 289–306.
[19] N.D. Cesare, D. Chamoret, M. Domaszewski, A new hybrid PSO algorithm based on a stochastic Markov chain modal, Adv. Eng. Softw. 90 (2015) 127–137.
[20] L. Wang, B. Yang, J. Orchard, Particle swarm optimization using dynamic tournament topology, Appl. Soft Comput. 48 (2016) 584–596.
[21] Q. Liu, W. Wei, H. Yuan, et al., Topology selection for particle swarm optimization, Inform. Sci. 363 (2016) 154–173.
[22] W. Sun, A. Lin, H. Yu, All-dimension neighborhood based particle swarm optimization with randomly selected neighbors, Inform. Sci. 405 (2017) 141–156.
[23] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (3) (2006) 281–295.
[24] A. Meng, Z. Li, H. Yin, S. Chen, Z. Guo, Accelerating particle swarm optimization using crisscross search, Inform. Sci. 329 (2016) 52–72.
[25] Q. Cui, Q. Li, G. Li, et al., Globally-optimal prediction-based adaptive mutation particle swarm optimization, Inform. Sci. 418 (2017) 186–217.
[26] H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, M. Ventresca, Enhancing particle warm optimization using generalized opposition-based learning, Inform. Sci. 181 (2011) 4699–4714.
[27] H. Huang, H. Qin, Z. Hao, Example-based learning particle swarm optimization for continuous optimization, Inform. Sci. 182 (2012) 125–138.
[28] L. Cao, L. Xu, E.D. Goodman, A neighbor-based learning particle swarm optimizer with short-term and long-term memory for dynamic optimization problems, Inform. Sci. 453 (2018) 463–485.
[29] J. Dong, L. Zhang, T. Xiao, A hybrid PSO/SA algorithm for bi-criteria stochastic line balancing with flexible task times and zoning constraints, J. Intell. Manuf. 29 (2018) 737–751.

[30] R. Wang, B. Chen, S. Qiu, Hazardous source estimation using an artificial neural network, particle swarm optimization and a simulated annealing algorithm, Atmosphere 9 (2018) http://dx.doi.org/10.3390/atmos9040119.
[31] J.H. Tam, C.O. Zhi, Z. Ismail, Inverse identification of elastic properties of composite materials using hybrid GA-ACO-PSO algorithm, Inverse. Probl. Sci. Eng. 26 (2018) 1432–1463.
[32] L.X. Wei, X. Li, R. Fan, A hybrid multi-objective particle swarm optimization algorithm based on R2 indicator, IEEE Access 6 (2018) 14710–14721.
[33] Y. Chen, L. Li, H. Peng, Particle swarm optimizer with two differential mutation, Appl. Soft. Comput. 61 (2017) 314–330.
[34] S. Cheng, H. Zhan, Z. Shu, An innovative hybrid multi-objective particle swarm optimization with or without constraints handling, Appl. Soft. Comput. 47 (2016) 370–388.
[35] L. Tong, X. Li, J. Hu, A PSO optimization scale-transformation stochastic-resonance algorithm with stability mutation operator, IEEE Access 6 (2018) 1167–1176.
[36] F. Wang, H. Zhang, K. Li, A hybrid particle swarm optimization algorithm using adaptive learning strategy, Inform. Sci. 436 (2018) 162–177.
[37] K. Gholami, E. Dehnavi, A modified particle swarm optimization algorithm for scheduling renewable generation in a micro-grid under load uncertainty, Appl. Soft Comput. 78 (2019) 496–514.
[38] SH. Wang, YZ. Li, HY. Yang, Self-adaptive mutation differential evolution algorithm based on particle swarm optimization, Appl. Soft Comput. 81 (2019) 1–21.
[39] N. Hansen, Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed, in: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, ACM, 2009, pp. 2389–2396.
[40] J.J. Liang, et al., Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization, Technical Report 201212, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China, 2013, Nanyang Technological University, Singapore.