Multiple adaptive strategies based particle swarm optimization algorithm

Bo Wei[a], Xuewen Xia*[b], Fei Yu[b], Yinglong Zhang[b], Xing Xu[b], Hongrun Wu[b], Ling Gui[b], Guoliang He[c]

[a]*School of Informatics Science and Technology, Zhejiang Sci-Tech University, Hangzhou, 310018, China*
[b]*College of Physics and Information Engineering, Minnan Normal University, Zhangzhou 363000, China*
[c]*School of Computer, Wuhan University, Wuhan 430072, China*

Abstract

Although particle swarm optimization algorithm (PSO) has displayed promising performance on many optimization problems, how to balance contradictions between the exploration and the exploitation and rationally allocate computational resource are two crucial problems need to be dealt with in PSO study. In this paper, a PSO variant based on multiple adaptive strategies (MAPSO) is proposed. To efficiently maintain the population diversity, the entire population is split into multiple swarms, which can be regrouped during the evolutionary process. In each generation, different particles in a swarm adaptively select their learning exemplars (ALE) according to the performance of the particles. Thus, different particles in the same swarm can perform distinct search behaviors in each generation, as well as the same particle can conduct various search behaviors in different generations. In addition, aiming to rationally utilize computational resource, an adaptive strategy for population size (APS) is introduced. In APS, the population can adaptively delete unfavorable particles and add promising particles during the evolutionary process. Extensive experiments based on CEC2013 and CEC2017 test suites verify the superior performance of the multiple adaptive strategies on balancing the exploration and exploitation abilities. Furthermore, the performance of the newly introduced strategies is also testified by a set of experiments.

Keywords: Particle swarm optimization, Multiple adaptive strategies,

Learning exemplars, Population size, Multiple swarms

1. Introduction

In recent years, bio-inspired computation has evolved as an interesting research field [1], and many swarm intelligence algorithms, such as particle swarm optimization algorithm (PSO) [3, 4] and honey bees algorithm [2], have proposed during the past decades. It is not only for its efficiency in finding an optimum, but also due to its affinity with natural social systems. PSO is a widely known EA proposed in 1995 [3, 4]. During the optimization process, each particle adjusts its flight direction and step-size relying on information extracted from past experience of itself and its neighbors. Although the search pattern of each particle is quite simple, the search behavior of the entire population is very complex and the swarm shows great intelligence owing to the cooperative behavior among particles. Due to the simplicity of implementation, PSO has been successfully applied in many real applications [5, 6, 7].

Extensive studies reveal that PSO's capability of finding a global optimum mainly depends upon two characteristics [8, 9], i.e., exploration and exploitation. However, there is a contradiction between the exploration and the exploitation. For example, the exploration is beneficial for multimodal functions, but it sacrifices performance on unimodal functions. On the contrary, the exploitation is more favorable for unimodal functions than multimodal functions. It is an intuitive idea that enhancing PSO's exploration (or exploitation) capability for a specific problem. Nevertheless, it is unrealistic to predefine search behaviors of PSO since it is very hard, if not impossible, to obtain essential characteristics of a given problem.

Aiming at the issue, many studies have been carried out during the past decades to obtain a tradeoff between the exploration and the exploitation [10, 11], and then to enhance the comprehensive performance of PSO. The common feature of these studies is keeping the population diversity and speeding up the convergence rate in the early and later evolutionary stages, respective-

ly. Moreover, rational allocating computational resources is another crucial issue when applying an algorithm in real applications. The most popular and simplest method is continual reducing population size [12] or increasing new individuals into the current population during the evolutionary process [13] while some conditions are satisfied. Some experiments verify that adaption strategies for population size offer a positive role in rational allocating computational resources [14, 15].

Inspired by the aforementioned research, a new PSO based on multiple adaptive strategies (MAPSO) is proposed in this paper. The main characteristics of MAPSO can be summarized as follows.

1. The entire population of MAPSO is randomly divided into multiple swarms in each generation. During the evolutionary process, particles in each swarm adaptive select their distinct learning exemplars depending on fitness values. Thus, not only different particles have different exemplars, but also the same particle can adaptively select different exemplars in different generations.
2. To rationally assign computational resources in different evolutionary stages, an adaptive adjustment for population size is introduced in MAPSO. With the adaptive adjustment, the population size can be altered in different fitness landscapes.
3. When adding new individuals into the current population, a simple differential evolution (DE) algorithm is applied to breed the individuals. Based on the process, merits of the two population-based heuristic algorithms can be organically integrated.

The rest of this paper is organized as follows. Section 2 presents the framework of the basic PSO and reviews some PSO modifications. Details of MAPSO algorithm are described in Section 3. Section 4 details the experimental verifications and comparisons using CEC2013 and CEC2017 test suites. Furthermore, performance of newly introduced strategies is testified by extensive experiments in Section 5. Finally, conclusions are presented in Section 6.

2. Related works

2.1. Canonical PSO

In the canonical PSO, each particle is regarded as a potential solution while the flight trajectory of a population is regarded as a continuous optimization process of PSO. In the t^{th} generation, the i^{th} particle is associated with two vectors, i.e., a position vector $\mathbf{X}_i^t = [x_{i,1}^t, x_{i,2}^t, ..., x_{i,D}^t]$ and a velocity vector $\mathbf{V}_i^t = [v_{i,1}^t, v_{i,2}^t, ..., v_{i,D}^t]$, where D represents the dimensions of a problem under study. During the evolutionary process, the i^{th} particle adjusts its flight trajectory based on two vectors, named as a personal historical best position $\mathbf{Pb}_i^t = [pb_{i,1}^t, pb_{i,2}^t, ..., pb_{i,D}^t]$ and its neighbor's historical best position $\mathbf{Nb}_i^t = [nb_{i,1}^t, nb_{i,2}^t, ..., nb_{i,D}^t]$. Update rules of $\mathbf{V}_i^t$ and $\mathbf{X}_i^t$ are defined as (1) and (2), respectively.

$$v_{i,j}^{t+1} = w \cdot v_{i,j}^t + c_1 \cdot r_{1,j} \cdot (pb_{i,j}^t - x_{i,j}^t) + c_2 \cdot r_{2,j} \cdot (nb_{i,j}^t - x_{i,j}^t) \tag{1}$$

$$x_{i,j}^{t+1} = x_{i,j}^t + v_{i,j}^{t+1} \tag{2}$$

where w represents an inertia weight denoting how much the previous velocity is preserved; c_1 and c_2 are two acceleration coefficients determining two learning weights for $\mathbf{Pb}_i^t$ and $\mathbf{Nb}_i^t$, respectively; $r_{1,j}$ and $r_{2,j}$ are two random numbers uniformly distributed in the interval [0, 1]; $x_{i,j}^t$ and $v_{i,j}^t$ represent the position and velocity in the j^{th} dimension of the i^{th} particle at generation t, respectively.

Generally speaking, three parts to the right of Eq. (1) are named as "inertia", "self-cognitive", and "social-learning", respectively.

2.2. Variants of PSO

PSO has been extensively studied since it was proposed due to its effectiveness and simplicity, and various PSO variants have been proposed in the last decades. Generally, PSO variants are generally divided into three categories, i.e., parameters tuning, learning exemplars selection, and hybridization strategy.

2.2.1. Parameters tuning

Since three important parameters in Eq.(1), i.e., w, c_1, and c_2, determine a particle's learning process, many research propose different settings of the parameters based on stability analysis [16]. Considering the population in different evolutionary stages should to satisfy various objective, e.g., the exploration or the exploitation, some scholars proposed different time-varying parameters for PSO. For instance, the most popular update rule of w is linear decreasing from 0.9 to 0.4 over the evolutionary state [17], which is still applied in many PSO variants now. Motivated by the generation-based time-varying adjustments of w, Ratnaweera *et al.* [10] further advocated a PSO with time-varying acceleration coefficients (HPSO-TVAC), where a larger c_1 and a smaller c_2 are set at the initial evolutionary stage and are gradually reversed during the search process.

Although the time-varying adjustments of the parameters are easy to be implemented, they cannot offer very reliable and comprehensive performance since different problems have their own distinct characteristics. Thus, to layout a more satisfactory tuning method for the parameters, various adaptive strategies are proposed in last decades [18]. Generally, the distribution of the population [19], the distance between different particles [20], and fitness of particles [21] are three common indexes to adaptive adjust the parameters in PSO. Furthermore, using other methods [22, 23] to adaptive tune the parameters also manifest a favorable performance. Extensive experiments indicate that these adaptation techniques endow PSO with very reliable performance, both in the unimodal and multimodal problems [24, 25], though the the process is complex.

2.2.2. Learning exemplars selection

According to selection strategies of learning exemplars, there are two popular topological structures, i.e., global version topological structure PSO (GPSO) and local version topological structure PSO (LPSO). Kennedy *et al.* [11] suggested that a sparse neighbor topology is conducive to complicated multimodal problems, while a dense neighbor topology is beneficial for simple unimodal problems.

To further improve the performance of PSO, many researchers proposed various adjustments for the topological structure intending to give a particle diversified selections of learning exemplars. For example, it is a popular strategy that using different methods to synthesize all (or a few of) particles as an promising exemplar for a specific particle [26, 27, 28]. The motivation of these type of improvements is that the generated exemplar could contain helpful information of many different particles based on an efficient integration strategy. However, variabilities of different problems cause that it is very hard to predefine a proper synthesize method to generate exemplars. Moreover, some researchers regard that it is improper to fix a proper neighbor topology to choose exemplars for a particle since many real applications are black-box problems and the optimization process is dynamic. Thus, quite a few dynamic selections of learning exemplar have been proposed in recent years [29]. For instance, various distance-based [30] and dynamic multi-swarms-based selection strategies for exemplars [31, 32, 33, 34] have attracted much attention of many scholars. Extensive experiments demonstrate that the adaptive and dynamic selecting process can improve the local searching ability and overcomes the premature phenomena of PSO to some extent.

2.2.3. Hybridization strategy

Applying a hybridization strategy is a popular way to enhance the performance of PSO on complex problems since different operators or algorithms involved in a hybrid PSO have their own merits [35]. For instance, the genetic operators [36, 37, 38] and some local searching strategies [39, 40, 41] are popular auxiliaries to improve the population diversity and to speed up the convergence rate, respectively. To take full advantage of some existing PSO variants, Lynn proposed an ensemble PSO (EPSO) [42], in which 5 PSO variants are integrated. Apart from these strategies, some scholars also pour attention into hybridizing PSO with other EAs due to that different EAs have displayed their own advantages on different real applications during the last few decades. For example, Xin *et al.* [43, 44] proposed a hybrid PSO based on differential evolution (DE)

algorithm and PSO. In [45], a hybrid PSO based on artificial bee colony (ABC) was proposed. The experiments of these type of hybrid PSOs indicate that a few deficiencies of the different basic components can be addressed by organic hybridization strategies, and the effectiveness of these hybrid algorithms has been verified by some real applications [46, 47].

It needs to be noted that some strategies that belong to different categories introduced above are simultaneously applied in a PSO variant in many cases [48, 49], though variants of PSO are classified into the three categories in this paper.

3. MAPSO

From Eq. (1) we can see that $\mathbf{Nb}_i^t$ plays a critical role since it is a main external knowledge provider for the i^{th} particle. In the canonical PSO, the i^{th} particle chooses $\mathbf{Nb}_i^t$ as its exemplar according to index numbers. In this case, the particle is incapable for timely adjusting its learning exemplars to meet different search objectives. Thus, to overcome the shortcomings of the index-based selection strategy, different selection strategies have been proposed during the last decades [50]. These studies verify that assigning different learning exemplars for a particle at different evolutionary states can yield more favorable performance than the static index-based strategy. Furthermore, multi-swarm mechanisms have been successfully applied in various EAs [31, 51] since they can yield positive performance in keeping population diversity.

In human society, we can find out a common phenomenon that different persons select their own exemplars according to their distinct capabilities. For instance, a more outstanding person in a specific field has more chance to (or prefer to) communicate with more distinguished persons in other fields. Thus, it seems to be a promising choice that each particle in PSO selects its own learning exemplars according to performance of it and the exemplars.

Moreover, rational distributing computational resources is a crucial issue for an efficient EA. A common method is that the population size continually re-

duces during the evolutionary process [12] or supplementing new individuals into the current population when stagnation is detected [13]. Extensive experimental results manifest that adaption strategies for population size play a positive role in EAs [13, 14, 15]. We regard that, for a specific problem, fitness landscapes in different search areas have their own distinctive characteristics. Generally, a relatively smaller population size is more suitable for simple fitness landscapes, while a larger population size is more conducive to complicated ones. Thus, dynamic adjusting the population size, i.e., both deleting and adding individuals, in the evolutionary process according to current fitness landscapes is a promising strategy to rationally allocate the computational resources.

Inspired by these observations, we propose a PSO algorithm based on multiple adaptive strategies (MAPSO) in this research. In MAPSO, the entire population is divided into multiple swarms. During the evolutionary process, particles in each swarm adaptively select their own candidate learning exemplars (ALE) according to the performance of them. Furthermore, an adaptive strategy for population size (APS) is adopted to rationally allocate computational resources, and then improve MAPSO's comprehensive performance.

3.1. Adaptive of learning exemplars (ALE)

During the evolutionary process, the entire population consists of multiple non-overlapping swarms with the same size. Note that, the number of particles in each swarm Np is set as 3 in this work. In each generation, 3 particles in a swarm can be categorized into three types, i.e., an elite particle, a mediocrity particle, and an inferior particle, based on their fitness. Accordingly, different types of particles choose their own learning exemplars intending to satisfy their own distinct search requirements.

Concretely, the elites dedicate to the exploitation aiming to enhance the solution accuracy, while the inferior particles are focusing on the exploration intending to find out more promising solutions in other regions. For the mediocrity particles, assigning a trade-off search behavior is a pleasurable choice. Hence, in each generation, the elite, mediocrity, and inferior particles in each swarm

are committed to exploitation, trade-off, and exploration abilities, respectively. Thus, particles in each swarm adaptively select their own learning exemplars to obtain much helpful information aiming to satisfy their own distinct search objectives.

In MAPSO, a particle only selects particles that belong to a favorable type as its candidate learning exemplars. Concretely, for an elite, candidate learning exemplars of it consists of all elites in all swarms. In this case, cooperations among the elites are advantageous to exploitation. On the contrary, candidate learning exemplars of an inferior, the primary task of which is keeping diversity, are particles in the entire population. For a mediocrity, elites and mediocrities in all swarms are regarded as its candidate learning exemplars. Under the condition, the exemplars can offer much positive knowledge and diverse information of elites and mediocrities, respectively, for the mediocrity. The adaptive of learning exemplars (ALE) for different particles can be demonstrated in Figure 1.

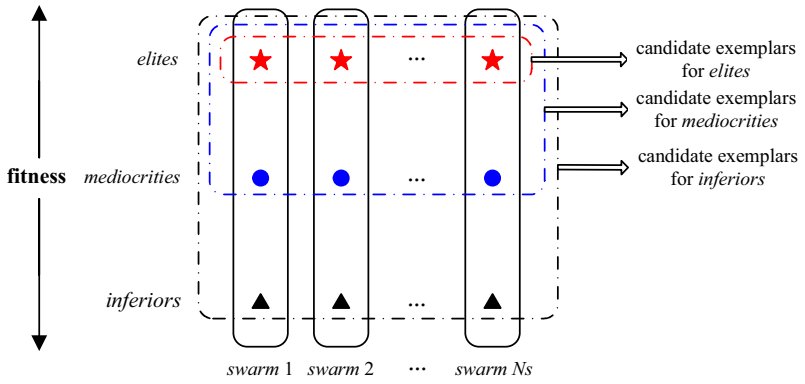


Figure 1: Candidate learning exemplars for different particles in each generation.

Based on ALE, not only different particles in the same swarm have their own distinct candidate exemplar in a generation, but also a particle has different exemplars in different generations.

After choosing some candidate learning exemplars, a particle i can perform its own learning model to update its velocity and position, and then conducts

its own search process. In this study, the i^{th} particle has two exemplars in t generation, i.e., $\mathbf{Nb}_i^t$ and $\mathbf{RNb}_i^t$. Note that $\mathbf{Nb}_i^t$ is the best-so-far position of the swarm that the i^{th} particle belongs to, while $\mathbf{RNb}_i^t = [rnb_{i,1}^t, rnb_{i,2}^t, ..., rnb_{i,D}^t]$ is a random selected neighbor particle from the i^{th} particle's candidate learning exemplars. The update rule can be described as (3).

$$v_{i,j}^{t+1} = w \cdot v_{i,j}^t + r_{1,j} \cdot (nb_{i,j}^t - x_{i,j}^t) + r_{2,j} \cdot (rnb_{i,j}^t - x_{i,j}^t) \tag{3}$$

Note that, to reduce the number of parameters and then to simplify the update rule of velocity, two acceleration coefficients applied in the canonical PSO (see Eq. (1)) are removed in MAPSO.

From (3) we can see that the two exemplars of an elite particle i have outstanding performance, in terms of fitness. Thus, the elite particle can pay more attention to exploitation. On the contrary, the randomly selected exemplar $\mathbf{RNb}_i^t$ for an inferior particle may have unsatisfied performance. As a result, the inferior particle may display promising characteristics on exploration. For a mediocrity particle in a swarm, two exemplars can help the particle to obtain a trade-off between exploration and exploitation.

Based on the above introduction, the particles' learning process in generation t is detailed as **Algorithm 1**.

3.2. Adaptive of population size (APS)

To rational assign computational resource, an adaptive adjustment of population size (APS) consisting of two processes (i.e., deleting particles and adding particles) is performed during the evolutionary process. The main idea of APS is that deleting some unfavorable particles from the current population while the fitness landscape is easy to be optimized. On the contrary, some pleasurable particles are added to the current population if the population cannot find out more promising solutions. Because it is impracticable that without limitation of population size when performing APS strategy, two predefined integers, i.e., N_{max} and N_{min}, are used to clamp the population size within the interval

Algorithm 1. ABS ( )

Input: t, Pop;
01: $N = |Pop|$; /* N is the population size of current population Pop. */
02: **For** i=1 to N
03: **If** $\mathbf{X}_i^t$ has the best fitness within the swarm it belongs to /* elite */
04: Select elites as candidate exemplars for $\mathbf{X}_i^t$ according to Figure 1;
05: **ElseIf** $\mathbf{X}_i^t$ has the worst fitness within the swarm it belongs to /* inferior */
06: Select all particles as candidate exemplars for $\mathbf{X}_i^t$ according to Figure 1;
07: **Else** /* mediocrity */
08: Select elites and mediocrities as candidate exemplars for $\mathbf{X}_i^t$ according to Figure 1;
09: **End if**
10: Update $\mathbf{V}_i^{t+1}$ and $\mathbf{X}_i^{t+1}$ based on Eqs. (3) and (2), respectively;
11: Evaluate $\mathbf{X}_i^{t+1}$; Update $\mathbf{Pb}_i^{t+1}$;
11: **End For**
12: Update $\mathbf{Gb}_i^{t+1}$; $t = t + 1$;
13: Randomly divide Pop into multiple swarms.
Output: Pop.

$[N_{min}, N_{max}]$. Details of the deleting particles and adding particles processes are presented as follows.

3.2.1. Deleting particles process

Generally speaking, when successive improvement generations (IG) of the global best particle ($\mathbf{X}_{gbest}^t$) is greater than a predefined threshold IG_{max}, we regard that the current fitness landscape is easy to be optimized, and deleting a few unfavorable particles from current population is propitious to save computational resources.

When deleting particles from the current population, some unfavorable particles are removed from the population. Considering that all swarms in MAPSO have the same size Np, the number of deleted particles (Nd) is a multiple of Np. Under the condition, the new population also can be divided into multiple swarms with the same size Np. Note that only if the size of the current population is greater than N_{min} does the deleting process need to be performed. After the deleting process, particles in the population need to be randomly divided into multiple swarms. The procedure of the deleting particles process is presented in **Algorithm 2**.

Algorithm 2. Deleting_Particles ( )

Input: Pop, IG, IG_{max}, N_{min};

01: $N = |Pop|$; // N is the current population size

02: **If** $IG \geq IG_{max}$ && $N > N_{min}$

03: IG=0;

04: Sort Pop satisfying $f(\mathbf{X}_{i_1}^t) \geq f(\mathbf{X}_{i_2}^t) \geq \ldots \geq f(\mathbf{X}_{i_N}^t)$;

05: Pop=$Pop \setminus \{\mathbf{X}_{i_1}^t, \mathbf{X}_{i_1}^t, \ldots \mathbf{X}_{i_{Nd}}^t\}$;

06: **End If**

Output: Pop, IG.

3.2.2. Adding particles process

When successive stagnation generations (SG) of $\mathbf{X}_{gbest}^t$ is larger than a predefined threshold SG_{max}, we think that the population cannot effectively deal with the current fitness landscape. Thus, a few favorable particles need to be added into the current population aiming to enhance the population's search capability.

When adding some new particles into the current population, a crux of the matter is how to generate new particles. The easiest way to deal with the matter is random generating some particles within the feasible space $\mathfrak{R}^D$. However, we regard that breeding new particles based on some elites of different swarms is a promising method since there is much helpful knowledge contained in the elites. Thus, in this work, the elites of all swarms saved in an archive A_E are utilized to generate a few particles that should be added into the current population. The maximum size of A_E is set as the number of swarms Ns at the initial evolutionary stage.

At the initial evolutionary process (i.e., $t = 1$), all elites $\mathbf{X}_{lbest_i}^t$ $(1 \leq i \leq Ns)$ are saved in A_E. Note that $\mathbf{X}_{lbest_i}^t$ is the best particle in the i^{th} swarm at generation t. Along with the process of evolutionary, some new elite solutions can be founded by the population. To save those newly emerged elites, A_E needs to be updated during the evolutionary process. Specially, only a new elite is not included in A_E can the elite be added into the archive. Meanwhile, the worst particle in A_E needs to be removed aiming to keep the size of A_E static. The update procedure of A_E is detailed in **Algorithm 3**.

Algorithm 3. Update_A_E ( )

Input: t, A_E, $\mathbf{X}^t_{lbest_i}$ $(1 \leq i \leq Ns)$;
If A_E==Ø /*Initialize A_E */
  A_E=$\{E_1, E_2, \ldots, E_{Ns}\}$=$\{\mathbf{X}^t_{lbest_1}, \mathbf{X}^t_{lbest_2}, \cdots, \mathbf{X}^t_{lbest_{Ns}}\}$;
Else /*Update A_E */
  For i=1 to Ns
    If $\mathbf{X}^t_{lbest_i} \notin A_E$
      $A_E = A_E \cup \mathbf{X}^t_{lbest_i}$;
    End If
  End For
End If
If $|A_E| > Ns$ /* $|A_E|$ is the size of A_E */
  Sort A_E satisfying $f(E_{i_1}) \leq f(E_{i_2}) \leq \ldots \leq f(E_{i_{|A_E|}})$;
  A_E=$A_E \setminus \{E_{i_{Ns+1}}, E_{i_{Ns+2}}, \ldots, E_{i_{|A_E|}}\}$;
End If
Output: A_E.

Based on the elites saved in A_E, new particles that need to be added into the current population can be generated. In this study, differential evolution algorithm (DE) [52], the performance of which depends on mutation, crossover, and selection operators as well as related control parameters, is used to breed the new particles. The procedure of the DE-based breeding process is introduced as follows.

In each generation t, a mutant vector $\mathbf{M}_i = (m_{i,1}, m_{i,2}, ..., m_{i,D})$ is created for each elite $\mathbf{E}_i$ (named as a target vector) by a mutation operator. Although there are a few mutation strategies [53, 54], due to the space limitation, only the common mutation strategy "DE/rand/1" is applied in this work, the details of which are described as Eq. (4).

$$\mathbf{M}_i = \mathbf{E}_{r_1} + F \cdot (\mathbf{E}_{r_2} - \mathbf{E}_{r_3}) \tag{4}$$

The indices r_1, r_2, and r_3 in Eq. (4) are mutually exclusive integers randomly selected from the set $\{1, 2, ..., Ns\} \setminus \{i\}$. The scaling factor F is a positive parameter used to scale the difference vectors $\mathbf{E}_{r_2} - \mathbf{E}_{r_3}$.

After generating the mutant vector $\mathbf{M}_i$, DE executes a crossover operator to generate a trial vector $\mathbf{U}_i = (u_{i,1}, u_{i,2}, ..., u_{i,D})$ based on each pair of $\mathbf{E}_i$ and $\mathbf{M}_i$.

Although there are many different crossover strategies, the binomial crossover is the easiest and widely accepted operator in many DE algorithms [39, 40, 54], which is defined as follows.

$$u_{i,j} = \begin{cases} m_{i,j}, & \text{if } rand_j(0,1) \leq CR \text{ or } j = j_{rand} \\ x_{i,j}, & \text{otherwise} \end{cases} \tag{5}$$

where $rand_j$ is a random number uniformly distributed in the range [0, 1]; the crossover rate CR is applied to determine the probability that $u_{i,j}$ is copied from $m_{i,j}$ or $x_{i,j}$; and j_{rand} is a randomly selected integer in the range [1, D].

When $u_{i,j}$ exceeds out of a feasible region, it needs to be reset within a feasible region by Eq. (6).

$$u_{i,j} = \begin{cases} min\{U_j, 2L_j - u_{i,j}\}, & \text{if } u_{i,j} < L_j \\ max\{L_j, 2U_j - u_{i,j}\}, & \text{if } u_{i,j} > U_j \end{cases} \tag{6}$$

where U_j and L_j are the upper and lower boundaries of the j^{th} dimension, respectively.

After the crossover operator, the better one between $\mathbf{E}_i$ and $\mathbf{U}_i$ is preserved to the next generation. Without loss of generality, minimization problems are considered in this paper. In this case, the selection operator can be described as follows.

$$\mathbf{E}_i = \begin{cases} \mathbf{E}_i, & \text{if } f(\mathbf{E}_i) < f(\mathbf{U}_i) \\ \mathbf{U}_i, & \text{otherwise} \end{cases} \tag{7}$$

where $f(x)$ is the function value of the vector x.

Based on the aforementioned description, the generating process of new particles can be described in **Algorithm 4**.

The procedure of adding the newly generated particles into the current population is presented in **Algorithm 5**. Similar to the deleting process, the number of added particles Na is also a multiple of Np. After adding a few particles to the current population, the population needs to be randomly divided into multiple swarms with the same size Np.

Algorithm 4. Generate_Particles ( )

Input: A_E, F, and CR;

01: **For** i=1 to $|A_E|$ /*$|A_E|$ is the size of archive A_E */

02: Randomly select $\mathbf{E}_{r_1}$, $\mathbf{E}_{r_2}$, and $\mathbf{E}_{r_3}$ from A_E;

03: Generate $\mathbf{M}_i$ according to (4); /* mutation */

04: Generate $\mathbf{U}_i$ according to (5) and (6); /* crossover */

05: Update $\mathbf{E}_i$ according to (7); /* selection */

06: **End For**

Output: $\mathbf{E}_i$=$[x_{i,1}, x_{i,2}, ..., x_{i,D}]$ $(1 \leq i \leq |A_E|)$.

Algorithm 5. Adding_Particles ( )

Input: t, Pop, A_E, SG, SG_{max}, N_{max};

01: **If** $SG \geq SG_{max}$

02: $SG = 0$; $N = |Pop|$;

03: Generate_Particles ( );

04: Sort particles in A_E satisfying $f(\mathbf{E}_{i_1} \leq \mathbf{E}_{i_2} \leq \cdots, \mathbf{E}_{i_{|A_E|}})$;

04: **If** $N < N_{max}$

05: $Pop = Pop \cup \mathbf{X}_i (1 \leq i \leq Na)$;

06: **Else**

07: Sort Pop satisfying $f(\mathbf{X}^t_{i_1}) \geq f(\mathbf{X}^t_{i_2}) \geq \ldots \geq f(\mathbf{X}^t_{i_N})$;

08: Replace $\mathbf{X}^t_{i_j}$ by $\mathbf{E}_{i_j} (1 \leq j \leq Na)$;

09: **End If**

10: Randomly divide Pop into multiple swarms with the same size Np;

11: **End If**

Output: Pop, SG.

3.3. Framework of MAPSO

By incorporating the aforementioned components into a multi-swarm PSO framework, the complete flowchart of MAPSO is described as Figure 2.

3.4. Computational complexity

It can be seen from Figure 2 that the main computational complexity of MAPSO occurs in the procedures ALE and APS. Thus, only the computational complexities of the procedures ALE and APS are analyzed as follows.

In ALE, the complexity of the Step B1 in Figure 2 is the same as the canonical PSO. In the Step B2 of ALE, 3 particles in each swarm are divided into 3 different types, the computational complexity of which is $O(1)$. Thus, the computational complexity of the division process of the entire population is $O(N)$. In the Step B3, the update process of A_E includes adding $\mathbf{X}^t_{lbest_i}$ into A_E (see

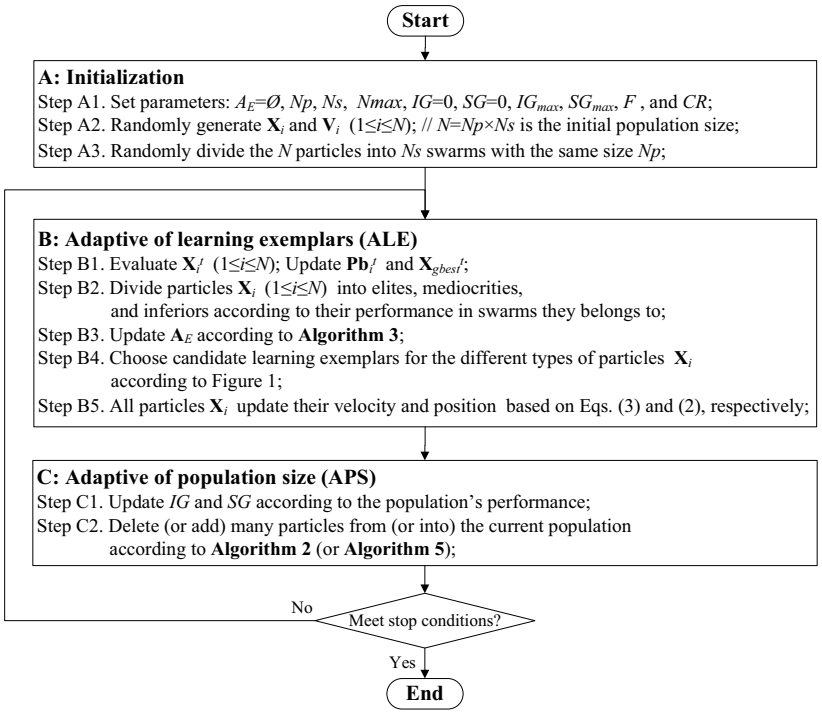


Figure 2: Flowchart of MAPSO algorithm.

line 04 - 08 in **Algorithm** 3) and sorting A_E (see line 10 - 12 in **Algorithm** 3), the complexities are $O(Ns^2)$ and $O(\log(Ns))$, respectively. Thus, the complexity of the update process of A_E is $O(Ns^2)$. In addition, the complexities of the Step B4 and the Step B5 are $O(1)$ and $O(N)$, respectively. Thus, the complexity of the procedure ALE is $O(N + Ns^2 + \log(Ns))$.

In APS, the complexity of the Step C1 in Figure 2 is $O(1)$. In the Step C2, the time consuming of the process deleting particles mainly focuses on the sort process (see line 04 in **Algorithm** 2). Thus, the complexity of it is $O(\log(Ns))$. In the process of adding particles, the time consuming is also caused by two sort processes (see line 04 and line 07 in **Algorithm** 5). Hence, the complexity of it is $O(\log(|AE|) + \log(N))$. As a result, the complexity of APS is $O(\log(Ns) + \log(|AE|) + \log(N))$.

Based on the aforementioned, we can see that the additional complexity

of MAPSO in each generation is $O(N + Ns^2 + \log(Ns))$ plus $O(\log(Ns) + \log(|AE|) + \log(N))$, i.e., $O(N + Ns^2 + \log(Ns) + \log(|AE|) + \log(N))$. Thus, if the algorithm is stopped after a fixed number of generation T, the overall computational complexity of MAPSO is $O(T \cdot (N + Ns^2 + \log(Ns) + \log(|AE|) + \log(N)))$. Note that, different operators have different time usages. Hence, to illustrate the time usage of MAPSO, an experiment of real time usages on different benchmark functions are presented in Section 4.3.2.

4. Experimental verification and comparisons

4.1. Experimental setup

4.1.1. Peer algorithms and benchmark functions

For a comparative analysis, 8 widely accepted PSO variants are selected in this study. Parameter settings of the selected peer algorithms summarized in Table 1 are the same as that in the corresponding papers.

Table 1: Basic Information of the Eight Peer Algorithms

Algorithm	Year	Parameters Settings
PSODDS [30]	2013	χ=0.7298, c_1=c_2=2.05
CCPSO-ISM [55]	2015	P=0.05, G=5, c=2.0, w=0.6
SRPSO [25]	2015	w_I=1.05, w_F=0.5, c_1=c_2=1.49445
HCLPSO [34]	2015	w=0.99∼0.2, c=3∼1.5, c_1=2.5∼0.5, c_2=0.5∼2.5, a=0, b=0.25
GLPSO [37]	2016	w=0.7298, c=1.49618, pm=0.01, sg=7
#EPSO [42]	2017	ensemble wPSO, CLPSO, FDR-PSO, HPSO-TVAC, and LIPS
*TAPSO [38]	2019	w=0.7298, p_c=0.5, p_m = 0.02, M=$N/4$
*AWPSO [20]	2019	$w = 0.9 \sim 0.4$, a=0.000035, b=0.5, c=0, d=1.5

There are too many parameters involved in EPSO. Due to the space limitation, more detailed information of the parameters setting one can refer to the corresponding literature.

* The two peer algorithms are published online in 2019.

To verify how MAPSO performs in different environments, two benchmark suites are chosen in this work. Firstly, three dimension cases (i.e., D=10, 30, and 50) of CEC2013 test suite [56] are tested intending to testify the scalability of the peer algorithms. Secondly, a more recent benchmark suite, i.e., CEC2017 test suite [57], also used to further verify the performance of all the algorithms on different benchmark functions.

4.2. Comparison on CEC2013 test suite

In the experiments conducted in this part, three dimension cases of the test functions in CEC2013 test suite, i.e., D=10, 30, and 50, are tested, while the maximum number of function evaluations ($MaxFEs$) is set to $10000 * D$ for all dimension cases. The performance of solution accuracy, in terms of mean value (*Mean*) and standard deviation (*Std.*), on the three dimension cases are detailed in Table 2, 3, and 4, respectively. The best results of the *Mean* on each problem among all algorithms are marked with shadow background.

4.2.1. Unimodal functions (f_1-f_5)

In the first set of experiments, 5 unimodal functions are studied. Table 2 shows that MAPSO achieves the best or almost the same as the best results on all of the $10D$ unimodal functions, in terms of solution quality ($Mean$), followed by GLPSO who yields the best performance on 3 out of the 5 unimodal functions. The results presented in Table 3 and Table 4 manifest that MAPSO also offers very promising performance on higher dimensionality cases since it displays the best results on the majority of the unimodal functions on both $30D$ and $50D$ cases. Furthermore, GLPSO, HCLPSO, and TAPSO also yield very favorable performance on the unimodal functions on three dimension cases. On the contrary, the performance of AWPSO decreases seriously along with the increase of dimensions.

4.2.2. Basic multimodal functions (f_6-f_{20})

In the second set of experiments, 15 basic multimodal functions are tested. The comparison results listed in Table 2 indicate that MAPSO achieves the best results on 5 basic multimodal functions measured by *Mean*, which is slightly better than CCPSO-ISM who yields the best result on 3 out of the 15 functions. SRPSO, HCLPSO, GLPSO, and EPSO display the same performance, in terms of the number the best *Mean* values. Along with the increase of dimensionality, MAPSO and TAPSO offer more favorable results. Specially, MAPSO achieves the best result on 3 and 6 basic multimodal functions on $30D$ and $50D$ cas-

Table 2: Comparison results of solution accuracy on CEC2013 test suite (D=10).

		PSODDS	CCPSO-ISM	SRPSO	HCLPSO	GLPSO	EPSO	TAPSO	AWPSO	MAPSO
f_1	*Mean*	1.78E-13(+)	4.46E-15(+)	9.81E-14(+)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	0.00E+00 (=)	1.29E-13(+)	0.00E+00
	Std.	7.69E-14	8.74E-15	1.12E-13	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.12E-13	0.00E+00
f_2	*Mean*	6.08E+04(=)	2.15E+06(+)	8.21E+04(+)	1.59E+05(+)	2.20E+04 (-)	1.34E+05(+)	1.73E+05(+)	1.33E+05(+)	3.17E+04
	Std.	7.83E+04	8.79E+05	7.53E+04	1.05E+05	1.07E+04	8.82E+04	1.68E+05	1.14E+05	2.22E+04
f_3	*Mean*	1.15E+08(=)	1.83E+08(+)	9.06E+07(=)	1.69E+05(+)	4.79E+06(+)	5.84E+05(+)	1.10E+06(+)	8.03E+08(+)	1.88E+04
	Std.	1.57E+08	1.19E+08	1.65E+08	2.18E+05	6.89E+06	7.32E+05	1.99E+06	1.34E+09	3.38E+04
f_4	*Mean*	8.34E+03(+)	1.44E+04(+)	1.98E+02(+)	1.31E+03(+)	1.94E+03(+)	5.25E+02(+)	3.60E+02(+)	7.32E+02(+)	5.22E+01
	Std.	5.57E+03	3.34E+03	1.33E+02	7.38E+02	1.35E+03	2.57E+02	3.53E+02	9.31E+02	4.83E+01
f_5	*Mean*	2.38E+01(+)	1.78E-13(+)	1.45E-13(+)	5.80E-14(+)	0.00E+00 (=)	8.69E-14(+)	7.58E-15(=)	3.05E+01(+)	0.00E+00
	Std.	3.93E+01	6.08E-14	4.77E-14	5.68E-14	0.00E+00	4.77E-14	1.41E-14	4.47E+01	0.00E+00
f_6	*Mean*	8.59E+00(=)	3.91E+00(-)	1.43E+01(+)	8.15E-01 (-)	6.17E+00(=)	9.83E-01(-)	5.38E+00(-)	2.14E+01(+)	7.13E+00
	Std.	4.93E+00	2.98E+00	8.78E+00	1.12E+00	4.57E+00	1.39E+00	5.13E+00	1.99E+01	3.89E+00
f_7	*Mean*	7.13E+01(+)	4.15E+01(+)	1.03E+01(+)	1.77E+00(+)	4.12E+00(+)	4.29E+00(+)	2.32E+00(+)	2.04E+01(+)	1.11E+00
	Std.	3.00E+01	8.21E+00	1.18E+01	1.03E+00	4.08E+00	2.68E+00	2.65E+00	1.82E+01	1.32E+00
f_8	*Mean*	2.04E+01(=)	2.03E+01 (-)	2.04E+01(=)	2.03E+01 (=)	2.04E+01(=)	2.04E+01(=)	2.04E+01(=)	2.03E+01 (-)	2.04E+01
	Std.	7.37E-02	5.72E-02	4.91E-02	6.39E-02	7.81E-02	4.63E-02	4.06E-02	5.03E-02	5.31E-02
f_9	*Mean*	6.15E+00(+)	5.88E+00(+)	2.46E+00(+)	2.42E+00(+)	2.71E+00(+)	3.09E+00(+)	2.51E+00(+)	3.48E+00(+)	1.73E+00
	Std.	1.18E+00	7.63E-01	8.12E-01	8.22E-01	9.84E-01	7.15E-01	8.91E-01	1.13E+00	8.70E-01
f_{10}	*Mean*	7.79E+00(+)	5.26E+00(+)	4.75E+00(+)	3.08E-01(+)	2.31E-01(+)	5.42E-01(+)	2.56E-01(+)	2.32E+01(+)	1.51E-01
	Std.	1.20E+01	1.40E+00	7.88E+00	1.49E-01	8.26E-02	3.02E-01	8.77E-02	2.74E+01	6.77E-02
f_{11}	*Mean*	1.25E+01(+)	2.88E-11(-)	4.04E+00(+)	1.17E-01(-)	1.95E-01(-)	8.92E-15 (-)	2.85E+00(+)	3.68E+00(+)	5.07E-01
	Std.	6.12E+00	4.85E-11	1.48E+00	2.11E-01	3.21E-01	1.50E-14	1.07E+00	1.81E+00	6.17E-01
f_{12}	*Mean*	3.32E+01(+)	3.33E+01(+)	9.74E+00(=)	8.53E+00(-)	1.15E+01(=)	8.22E+00 (-)	9.17E+00(-)	1.62E+01(=)	1.08E+01
	Std.	1.37E+01	6.40E+00	4.11E+00	2.76E+00	4.60E+00	2.56E+00	3.30E+00	5.32E+00	3.54E+00
f_{13}	*Mean*	5.79E+01(+)	3.81E+01(+)	1.32E+01 (=)	1.75E+01(=)	2.67E+01(+)	1.41E+01(=)	1.40E+01(=)	2.11E+01(=)	1.58E+01
	Std.	1.49E+01	6.85E+00	4.55E+00	5.36E+00	8.18E+00	4.67E+00	4.88E+00	4.49E+00	6.47E+00
f_{14}	*Mean*	5.39E+02(+)	9.88E+01(+)	3.26E+02(+)	4.41E+01(+)	8.18E-01 (-)	2.73E+00(-)	2.90E+02(+)	3.09E+02(+)	6.27E+00
	Std.	1.83E+02	6.43E+01	1.12E+02	5.55E+01	1.03E+00	1.98E+00	1.20E+02	1.23E+02	5.16E+00
f_{15}	*Mean*	1.11E+03(+)	7.99E+02(+)	3.47E+02 (-)	5.25E+02(-)	6.71E+02(=)	5.57E+02(=)	4.22E+02(-)	6.14E+02(=)	6.17E+02
	Std.	2.09E+02	1.31E+02	1.60E+02	1.48E+02	2.55E+02	1.65E+02	1.79E+02	1.63E+02	1.70E+02
f_{16}	*Mean*	5.13E-01(=)	3.30E-01(-)	1.14E+00(+)	4.02E-01(=)	3.09E-01 (-)	1.06E+00(+)	1.13E+00(+)	9.02E-01(+)	4.58E-01
	Std.	1.82E-01	6.36E-02	1.47E-01	1.40E-01	1.58E-01	1.51E-01	1.50E-01	2.18E-01	1.63E-01
f_{17}	*Mean*	2.43E+01(+)	8.83E+00 (-)	1.45E+01(+)	9.81E+00(-)	1.08E+01(-)	1.27E+01(+)	1.63E+01(+)	1.28E+01(+)	1.10E+01
	Std.	4.70E+00	2.51E+00	2.00E+00	1.10E+00	3.04E-01	1.01E+00	2.23E+00	2.44E+00	3.87E-01
f_{18}	*Mean*	3.18E+01(+)	3.58E+01(+)	3.39E+01(+)	2.08E+01(+)	2.13E+01(+)	3.00E+01(+)	2.96E+01(+)	2.45E+01(+)	1.65E+01
	Std.	6.95E+00	4.96E+00	4.77E+00	3.91E+00	4.41E+00	7.08E+00	5.99E+00	5.88E+00	1.95E+00
f_{19}	*Mean*	1.00E+00(+)	2.05E-01 (-)	7.67E-01(=)	3.59E-01(-)	6.00E-01(-)	6.15E-01(-)	6.08E-01(-)	8.35E+00(+)	6.92E-01
	Std.	3.24E-01	6.39E-02	1.73E-01	8.34E-02	1.21E-01	1.16E-01	1.75E-01	1.44E+01	1.74E-01
f_{20}	*Mean*	3.73E+00(+)	3.67E+00(+)	2.56E+00(+)	2.31E+00(+)	2.97E+00(+)	2.30E+00(+)	2.23E+00(=)	2.73E+00(+)	2.07E+00
	Std.	4.62E-01	2.80E-01	5.80E-01	4.00E-01	4.51E-01	3.96E-01	5.81E-01	6.97E-01	4.61E-01
f_{21}	*Mean*	3.00E+02(=)	1.46E+02 (-)	3.00E+02(=)	2.61E+02(-)	2.90E+02(=)	2.05E+02(-)	3.00E+02(=)	3.00E+02(=)	3.00E+02
	Std.	0.00E+00	9.94E+01	0.00E+00	6.31E+01	1.88E+01	1.03E+02	0.00E+00	3.41E-14	0.00E+00
f_{22}	*Mean*	7.33E+02(+)	1.53E+02(+)	3.59E+02(+)	5.91E+01(=)	6.65E+01(=)	4.37E+01 (-)	3.19E+02(+)	3.60E+02(+)	7.59E+01
	Std.	2.48E+02	5.72E+01	1.25E+02	5.80E+01	5.65E+01	2.26E+01	1.17E+02	1.88E+02	5.03E+01
f_{23}	*Mean*	1.20E+03(+)	1.22E+03(+)	5.26E+02 (=)	6.65E+02(=)	9.32E+02(+)	6.10E+02(=)	6.09E+02(=)	8.57E+02(+)	5.66E+02
	Std.	2.89E+02	1.54E+02	2.32E+02	1.86E+02	3.25E+02	2.10E+02	3.10E+02	2.34E+02	2.50E+02
f_{24}	*Mean*	2.20E+02(+)	1.53E+02(-)	1.94E+02(=)	1.40E+02(-)	2.10E+02(+)	1.26E+02 (-)	2.04E+02(=)	2.18E+02(+)	2.02E+02
	Std.	3.81E+00	1.27E+01	2.31E+01	2.99E+01	4.36E+00	1.91E+01	8.84E+00	4.33E+00	6.24E+00
f_{25}	*Mean*	2.14E+02(+)	1.94E+02(-)	2.08E+02(+)	1.81E+02 (-)	2.04E+02(=)	1.94E+02(=)	2.04E+02(=)	2.06E+02(=)	2.03E+02
	Std.	1.14E+01	2.53E+01	5.94E+00	3.08E+01	6.82E+00	1.89E+01	8.39E+00	3.84E+00	3.41E+00
f_{26}	*Mean*	1.93E+02(=)	1.47E+02(-)	1.26E+02(-)	1.36E+02(-)	1.66E+02(=)	1.13E+02 (-)	1.29E+02(-)	1.88E+02(=)	1.86E+02
	Std.	1.35E+01	1.39E+01	2.24E+01	3.53E+01	6.11E+01	5.85E+00	5.76E+01	6.46E+01	2.32E+01
f_{27}	*Mean*	5.30E+02(+)	3.98E+02(+)	4.20E+02(+)	3.23E+02(+)	3.59E+02(+)	3.26E+02(+)	3.02E+02 (=)	4.56E+02(+)	3.04E+02
	Std.	7.43E+01	3.51E+00	8.20E+01	1.71E+01	4.85E+01	1.31E+01	7.43E+01	6.36E+01	7.09E+00
f_{28}	*Mean*	3.20E+02(=)	1.95E+02 (-)	3.22E+02(+)	2.65E+02(-)	3.85E+02(+)	2.02E+02(-)	2.41E+02(-)	3.01E+02(=)	2.92E+02
	Std.	9.83E+01	1.10E+02	4.61E+01	5.81E+01	1.17E+02	9.99E+01	5.77E+01	5.47E+01	1.51E+01
(#)	+	20	17	18	11	12	12	12	20	
(#)	-	0	11	2	11	6	10	6	1	

es, respectively, while TAPSO offers the most favorable performance on 5 and 3 multimodal functions on the two dimension cases, respectively. Note that HCLPSO also displays very promising performance on the basic multimodal

Table 3: Comparison results of solution accuracy on CEC2013 test suite (D=30).

		PSODDS	CCPSO-ISM	SRPSO	HCLPSO	GLPSO	EPSO	TAPSO	AWPSO	MAPSO
f_1	*Mean*	8.94E+01(+)	2.27E-13(+)	1.19E+02(+)	3.34E-13(+)	4.46E-15(=)	2.50E-13(+)	3.79E-15(=)	3.79E+03(+)	0.00E+00
	Std.	1.54E+02	0.00E+00	1.87E+02	1.13E-13	8.74E-15	4.02E-14	1.62E-13	2.25E+03	0.00E+00
f_2	*Mean*	1.24E+06(=)	1.00E+07(+)	3.11E+06(+)	1.34E+06(=)	3.42E+05(-)	1.74E+05 (-)	1.22E+06(-)	1.80E+07(+)	1.49E+06
	Std.	1.32E+06	1.95E+06	2.37E+06	5.94E+05	1.42E+05	5.01E+04	2.63E+06	1.97E+07	9.77E+05
f_3	*Mean*	6.64E+09(+)	7.70E+08(+)	3.95E+09(+)	3.63E+07(+)	1.63E+07(=)	8.16E+07(+)	1.80E+06 (-)	3.32E+10(+)	1.46E+07
	Std.	6.30E+09	3.48E+08	4.92E+09	3.50E+07	1.28E+07	1.00E+08	1.94E+06	2.49E+10	1.31E+07
f_4	*Mean*	5.99E+03(+)	3.32E+04(+)	2.14E+02(+)	2.45E+03(+)	1.42E+03(+)	1.72E+02(+)	1.79E+02(+)	3.44E+03(+)	8.14E+01
	Std.	4.58E+03	5.06E+03	8.55E+01	7.69E+02	8.83E+02	1.00E+02	1.09E+02	4.38E+03	8.64E+01
f_5	*Mean*	6.50E+01(+)	2.56E-13(+)	1.21E+02(+)	3.63E-13(+)	1.07E-13(+)	2.79E-13(+)	4.21E-13(+)	2.25E+03(+)	6.91E-14
	Std.	6.69E+01	4.32E-14	1.58E+02	6.44E-14	1.26E-14	6.03E-14	1.21E-13	2.06E+03	5.42E-14
f_6	*Mean*	6.09E+01(+)	2.11E+01(+)	7.01E+01(+)	2.75E+01(+)	2.11E+01(=)	1.16E+01 (-)	2.63E+01(+)	1.97E+02(+)	1.69E+01
	Std.	3.18E+01	6.29E+00	1.88E+01	9.58E+00	1.25E+01	3.43E+00	2.25E+01	1.10E+02	1.55E+00
f_7	*Mean*	1.03E+02(+)	1.03E+02(+)	4.48E+01(+)	1.85E+01(=)	2.57E+01(+)	3.30E+01(+)	3.83E+00 (-)	1.08E+02(+)	1.87E+01
	Std.	2.22E+01	1.35E+01	1.98E+01	4.89E+00	1.31E+01	9.22E+00	2.70E+00	4.16E+01	7.53E+00
f_8	*Mean*	2.09E+01 (=)	2.09E+01 (-)	2.10E+01(=)	2.09E+01 (=)	2.09E+01 (=)	2.09E+01 (=)	2.09E+01 (-)	2.09E+01 (=)	2.10E+01
	Std.	4.79E-02	4.54E-02	2.91E-02	3.60E-02	6.38E-02	5.02E-02	2.65E-02	3.77E-02	4.00E-02
f_9	*Mean*	2.59E+01(+)	3.03E+01(+)	1.79E+01(+)	2.17E+01(+)	1.59E+01(+)	2.24E+01(+)	9.81E+00 (-)	2.02E+01(+)	1.22E+01
	Std.	2.20E+00	1.38E+00	2.94E+00	3.35E+00	2.54E+00	2.99E+00	2.13E+00	2.24E+00	2.35E+00
f_{10}	*Mean*	7.67E+01(+)	1.98E+00(+)	8.94E+01(+)	2.90E-01(+)	1.32E-01(=)	1.39E-01(=)	1.02E-01 (=)	4.99E+02(+)	1.10E-01
	Std.	5.15E+01	2.93E-01	7.07E+01	1.13E-01	5.41E-02	6.28E-02	4.89E-02	3.03E+02	4.61E-02
f_{11}	*Mean*	8.34E+01(+)	8.14E-14 (-)	3.80E+01(+)	1.03E-13(-)	7.80E-02(-)	1.14E-13(-)	1.12E+01(-)	1.02E+02(+)	1.22E+01
	Std.	2.34E+01	2.79E-14	7.99E+00	2.87E-14	1.44E-01	2.45E-14	3.81E+00	3.38E+01	4.72E+00
f_{12}	*Mean*	1.65E+02(+)	2.57E+02(+)	5.59E+01(+)	5.49E+01(+)	4.73E+01(=)	5.95E+01(+)	3.89E+01 (-)	1.20E+02(+)	4.43E+01
	Std.	3.73E+01	3.24E+01	1.49E+01	1.34E+01	1.52E+01	1.46E+01	1.98E+01	3.69E+01	9.23E+00
f_{13}	*Mean*	2.66E+02(+)	2.77E+02(+)	1.20E+02(+)	1.46E+02(+)	1.13E+02(+)	1.07E+02(+)	1.06E+02(+)	1.63E+02(+)	9.37E+01
	Std.	4.57E+01	1.90E+01	2.14E+01	1.07E+01	2.64E+01	3.01E+01	5.10E+01	2.74E+01	2.22E+01
f_{14}	*Mean*	1.99E+03(+)	6.89E+02(+)	1.34E+03(+)	1.74E+01(-)	3.35E+00(-)	3.21E+01(-)	1.39E+03(+)	1.85E+03(+)	1.71E+02
	Std.	4.80E+02	1.17E+02	2.12E+02	2.81E+01	1.88E+00	1.07E+01	4.35E+02	4.58E+02	8.86E+01
f_{15}	*Mean*	4.10E+03(+)	3.87E+03(+)	3.06E+03 (=)	3.99E+03(+)	3.81E+03(+)	3.62E+03(+)	5.27E+03(+)	3.50E+03(=)	3.30E+03
	Std.	3.86E+02	2.45E+02	7.06E+02	2.98E+02	4.53E+02	4.12E+02	1.72E+03	5.42E+02	6.41E+02
f_{16}	*Mean*	8.89E-01(-)	8.33E-01(-)	2.51E+00(+)	1.63E+00(+)	4.84E-01 (-)	1.70E+00(+)	2.40E+00(+)	2.07E+00(+)	1.33E+00
	Std.	4.29E-01	1.81E-01	2.29E-01	1.20E-01	2.27E-01	2.93E-01	2.44E-01	3.26E-01	3.87E-01
f_{17}	*Mean*	1.06E+02(+)	3.62E+01(-)	6.17E+01(+)	3.07E+01(-)	3.20E+01(-)	3.65E+01(-)	8.16E+01(+)	8.65E+01(+)	5.06E+01
	Std.	1.42E+01	1.33E+00	7.35E+00	8.08E-02	4.31E-01	2.35E+00	4.43E+01	3.58E+01	5.39E+00
f_{18}	*Mean*	1.69E+02(+)	1.73E+02(+)	1.98E+02(+)	1.03E+02(+)	7.63E+01(+)	8.07E+01(+)	1.93E+02(+)	1.71E+02(+)	6.79E+01
	Std.	3.26E+01	1.69E+01	2.06E+01	2.45E+01	1.13E+01	1.40E+01	8.48E+00	4.36E+01	1.39E+01
f_{19}	*Mean*	1.04E+01(+)	1.26E+00(-)	3.88E+00(=)	1.38E+00(-)	1.71E+00(-)	1.83E+00(-)	2.58E+00(-)	2.30E+03(-)	3.15E+00
	Std.	5.46E+00	4.16E-01	1.53E+00	2.20E-01	2.21E-01	2.78E-01	5.18E-01	3.14E+03	6.71E-01
f_{20}	*Mean*	1.41E+01(+)	1.47E+01(+)	1.38E+01(+)	1.09E+01(+)	1.14E+01(+)	1.10E+01(+)	1.18E+01(+)	1.48E+01(+)	9.97E+00
	Std.	8.68E-01	2.33E-01	1.31E+00	5.29E-01	6.85E-01	6.39E-01	1.20E+00	3.76E-01	5.43E-01
f_{21}	*Mean*	3.38E+02(+)	1.73E+02 (-)	3.44E+02(+)	2.58E+02(-)	3.13E+02(=)	2.26E+02(-)	3.10E+02(=)	5.26E+02(+)	2.92E+02
	Std.	8.31E+01	3.85E+01	7.75E+01	4.66E+01	4.58E+01	4.35E+01	8.01E+01	1.66E+02	5.06E+01
f_{22}	*Mean*	2.38E+03(+)	6.66E+02(+)	1.62E+03(+)	1.10E+02(-)	1.08E+02(-)	1.48E+02(-)	1.02E+02 (-)	2.33E+03(+)	2.19E+02
	Std.	4.45E+02	1.12E+02	3.38E+02	1.05E+01	2.15E+01	2.13E+01	2.83E+01	5.87E+02	6.47E+01
f_{23}	*Mean*	5.00E+03(+)	4.92E+03(+)	3.29E+03 (=)	4.10E+03(+)	3.91E+03(+)	4.00E+03(+)	4.79E+03(+)	4.13E+03(+)	3.44E+03
	Std.	7.40E+02	4.39E+02	5.42E+02	4.16E+02	5.98E+02	5.24E+02	2.09E+03	4.63E+02	5.71E+02
f_{24}	*Mean*	2.76E+02(+)	2.87E+02(+)	2.60E+02(+)	2.29E+02 (=)	2.37E+02(+)	2.47E+02(+)	2.33E+02(=)	2.73E+02(+)	2.26E+02
	Std.	6.59E+00	6.37E+00	1.16E+01	5.99E+00	1.01E+01	7.79E+00	1.84E+01	7.35E+00	8.91E+00
f_{25}	*Mean*	3.00E+02(+)	3.14E+02(+)	2.89E+02(+)	2.68E+02(+)	2.64E+02(+)	2.85E+02(+)	2.62E+02(=)	2.93E+02(+)	2.60E+02
	Std.	8.00E+00	5.10E+00	1.07E+01	1.12E+01	5.87E+00	7.18E+00	7.22E+00	8.17E+00	7.14E+00
f_{26}	*Mean*	2.47E+02(+)	2.01E+02(=)	2.76E+02(+)	2.00E+02 (=)	2.62E+02(+)	2.07E+02(=)	2.06E+02(=)	3.25E+02(+)	2.00E+02
	Std.	6.67E+01	1.98E-01	6.88E+01	2.39E-02	6.55E+01	1.30E+01	4.85E+01	4.94E+01	3.63E-01
f_{27}	*Mean*	9.89E+02(+)	6.82E+02(+)	7.78E+02(+)	6.09E+02(+)	6.76E+02(+)	7.30E+02(+)	4.81E+02 (-)	8.94E+02(+)	5.33E+02
	Std.	8.91E+01	3.42E+02	9.86E+01	1.44E+02	8.47E+01	1.49E+02	1.34E+02	6.62E+01	8.32E+01
f_{28}	*Mean*	1.11E+03(+)	2.51E+02 (-)	6.40E+02(+)	3.00E+02(=)	3.53E+02(=)	2.84E+02(-)	2.89E+02(=)	1.89E+03(+)	3.00E+02
	Std.	4.39E+02	7.30E+01	4.07E+02	9.79E-11	1.24E+02	2.89E+01	4.62E+01	5.25E+02	3.13E-13
(#)	+	25	21	24	16	13	16	11	25	
(#)	-	1	7	0	6	7	9	10	1	

functions on $50D$ case. However, CCPSO-ISM does not display very reliable performance on the higher dimensionality case though it yields a favorable performance on $10D$ case. Concretely, it cannot offer the best result on the basic

Table 4: Comparison results of solution accuracy on CEC2013 test suite (D=50).

		PSODDS	CCPSO-ISM	SRPSO	HCLPSO	GLPSO	EPSO	TAPSO	AWPSO	MAPSO
f_1	*Mean*	3.67E+02(+)	4.50E-13(+)	3.85E+02(+)	7.04E-13(+)	1.96E-13(+)	3.30E-13(+)	2.03E-13(=)	4.12E+03(+)	8.47E-14
	Std.	4.46E+02	8.74E-15	5.13E+02	8.04E-14	5.38E-14	1.13E-13	7.28E-14	1.84E+03	1.06E-13
f_2	*Mean*	1.02E+06(-)	2.23E+07(+)	3.20E+06(+)	2.65E+06(=)	9.03E+05 (-)	7.60E+06(+)	1.22E+07(+)	1.97E+07(+)	2.31E+06
	Std.	4.29E+05	3.87E+06	1.33E+06	6.49E+05	2.63E+05	3.12E+06	6.74E+06	1.27E+07	1.20E+06
f_3	*Mean*	5.74E+09(+)	3.52E+09(+)	3.07E+09(+)	2.32E+08(+)	8.75E+07(+)	4.45E+08(+)	3.79E+07(=)	2.95E+10(+)	3.67E+07
	Std.	4.65E+09	1.40E+09	3.15E+09	1.67E+08	8.56E+07	2.97E+08	6.50E+07	1.79E+10	2.39E+07
f_4	*Mean*	3.74E+03(+)	4.48E+04(+)	2.51E+02(+)	2.13E+03(+)	2.17E+03(+)	2.47E+03(+)	6.87E+02(+)	5.58E+03(+)	2.73E+01
	Std.	3.70E+03	4.29E+03	7.31E+01	6.28E+02	1.38E+03	5.62E+02	1.62E+02	4.96E+03	2.39E+01
f_5	*Mean*	1.09E+02(+)	4.64E-13(+)	1.86E+02(+)	8.02E-13(+)	1.34E-13(=)	6.38E-13(+)	3.06E-13(=)	1.92E+03(+)	1.20E-13
	Std.	6.86E+01	3.70E-14	2.19E+02	1.25E-13	3.30E-14	1.20E-13	1.79E-13	9.06E+02	1.26E-14
f_6	*Mean*	8.48E+01(+)	4.49E+01(+)	7.38E+01(+)	4.46E+01(+)	4.91E+01(+)	4.44E+01(+)	4.41E+01(=)	2.41E+02(+)	4.37E+01
	Std.	3.03E+01	1.43E+00	3.03E+01	6.07E-01	9.30E+00	6.02E-01	5.02E+00	9.35E+01	4.31E-01
f_7	*Mean*	1.13E+02(+)	1.29E+02(+)	5.42E+01(+)	4.24E+01(+)	4.59E+01(+)	6.45E+01(+)	2.68E+01 (-)	1.04E+02(+)	3.65E+01
	Std.	1.55E+01	9.38E+00	2.07E+01	8.86E+00	9.94E+00	7.72E+00	6.33E+00	2.53E+01	1.06E+01
f_8	*Mean*	2.11E+01 (=)	2.11E+01 (=)	2.11E+01 (=)	2.11E+01 (=)	2.11E+01 (=)	2.11E+01 (=)	2.11E+01 (=)	2.11E+01 (=)	2.11E+01
	Std.	3.39E-02	3.89E-02	3.47E-02	2.86E-02	5.49E-02	2.83E-02	3.38E-02	3.54E-02	2.70E-02
f_9	*Mean*	4.81E+01(+)	5.51E+01(+)	2.79E+01(=)	4.16E+01(+)	3.16E+01(+)	4.86E+01(+)	2.66E+01 (-)	4.12E+01(+)	2.83E+01
	Std.	4.17E+00	2.17E+00	3.27E+00	5.35E+00	3.36E+00	3.44E+00	4.34E+00	2.49E+00	3.87E+00
f_{10}	*Mean*	1.34E+02(+)	6.29E+00(+)	1.38E+02(+)	2.44E-01(+)	1.79E-01(+)	1.18E-01(=)	9.05E-02(-)	9.77E+02(+)	9.87E-02
	Std.	1.17E+02	1.32E+00	1.08E+02	1.17E-01	5.74E-02	5.40E-02	4.27E-02	5.18E+02	4.49E-02
f_{11}	*Mean*	1.51E+02(+)	8.05E+00(-)	8.70E+01(+)	2.31E-12(-)	4.68E-14 (-)	1.95E-02(-)	6.47E+01(+)	1.55E+02(+)	1.32E+01
	Std.	2.82E+01	2.47E+00	1.63E+01	2.73E-12	1.65E-14	3.83E-02	1.09E+01	4.17E+01	3.14E+00
f_{12}	*Mean*	3.26E+02(+)	4.61E+02(+)	1.09E+02(+)	1.27E+02(+)	1.08E+02(+)	1.85E+02(+)	1.14E+02(+)	2.20E+02(+)	8.54E+01
	Std.	5.56E+01	4.41E+01	2.48E+01	1.66E+01	2.15E+01	6.74E+01	4.04E+01	3.84E+01	1.56E+01
f_{13}	*Mean*	5.56E+02(+)	4.88E+02(+)	2.44E+02(+)	2.48E+02(+)	2.38E+02(+)	2.03E+02(=)	2.11E+02(+)	3.56E+02(+)	1.62E+02
	Std.	6.91E+01	3.09E+01	3.42E+01	4.01E+01	4.00E+01	5.19E+01	4.89E+01	4.36E+01	2.68E+01
f_{14}	*Mean*	3.99E+03(+)	1.56E+03(+)	2.73E+03(+)	3.24E+00 (-)	4.64E+00(-)	3.03E+01(-)	2.20E+03(+)	3.41E+03(+)	7.48E+01
	Std.	5.99E+02	2.18E+02	4.76E+02	1.14E+00	1.99E+00	6.73E+00	4.84E+02	5.15E+02	4.66E+01
f_{15}	*Mean*	7.53E+03(+)	7.69E+03(+)	7.54E+03(=)	6.99E+03(=)	7.34E+03(=)	7.05E+03(=)	6.75E+03 (-)	8.07E+03(+)	7.01E+03
	Std.	7.21E+02	4.62E+02	1.81E+03	6.63E+02	6.92E+02	5.63E+02	2.34E+03	1.50E+03	7.02E+02
f_{16}	*Mean*	8.35E-01(-)	1.52E+00(=)	3.38E+00(+)	1.80E+00(=)	6.91E-01 (-)	1.98E+00(+)	3.28E+00(+)	2.73E+00(+)	1.65E+00
	Std.	2.06E-01	2.84E-01	1.96E-01	2.85E-01	3.02E-01	2.44E-01	2.61E-01	3.08E-01	4.01E-01
f_{17}	*Mean*	2.27E+02(+)	7.81E+01(=)	1.27E+02(+)	5.13E+01 (-)	5.28E+01(-)	6.98E+01(-)	1.41E+02(+)	1.75E+02(+)	7.81E+01
	Std.	3.55E+01	4.84E+00	1.06E+01	1.07E-01	5.77E-01	2.10E+00	1.50E+01	4.30E+01	6.99E+00
f_{18}	*Mean*	3.33E+02(+)	3.20E+02(+)	3.96E+02(+)	1.73E+02(+)	1.44E+02(=)	2.48E+02(+)	3.40E+02(+)	3.65E+02(+)	1.37E+02
	Std.	7.59E+01	2.38E+01	4.43E+01	2.80E+01	1.65E+01	1.17E+02	6.46E+01	8.96E+01	4.55E+01
f_{19}	*Mean*	1.86E+01(+)	4.14E+00(-)	7.18E+01(=)	2.13E+00 (-)	2.80E+00(-)	4.64E+00(=)	7.34E+00(+)	1.92E+04(+)	4.72E+00
	Std.	7.86E+00	1.04E+00	1.21E+02	3.53E-01	3.88E-01	6.08E-01	1.13E+00	2.73E+04	8.37E-01
f_{20}	*Mean*	2.35E+01(+)	2.45E+01(+)	2.20E+01(+)	2.00E+01(+)	2.06E+01(+)	2.02E+01(+)	2.09E+01(+)	2.32E+01(+)	1.88E+01
	Std.	1.01E+00	1.48E-01	1.25E+00	8.33E-01	7.59E-01	8.00E-01	7.58E-01	1.75E+00	7.68E-01
f_{21}	*Mean*	8.72E+02(=)	2.49E+02(-)	8.13E+02(=)	2.39E+02 (-)	8.51E+02(=)	2.51E+02(-)	8.00E+02(=)	1.18E+03(+)	8.90E+02
	Std.	2.41E+02	8.59E+01	2.52E+02	4.70E+01	2.28E+02	6.19E+01	2.39E+02	2.59E+02	2.52E+02
f_{22}	*Mean*	5.27E+03(+)	1.77E+03(+)	3.60E+03(+)	3.47E+01 (-)	4.25E+01(-)	6.52E+01(-)	2.94E+03(+)	4.59E+03(+)	9.44E+01
	Std.	8.27E+02	3.47E+02	6.43E+02	2.19E+01	3.72E+01	3.01E+01	4.88E+02	7.84E+02	4.31E+01
f_{23}	*Mean*	9.43E+03(+)	9.83E+03(+)	7.32E+03(+)	8.23E+03(+)	8.21E+03(+)	8.71E+03(+)	8.13E+03(+)	8.82E+03(+)	6.69E+03
	Std.	1.07E+03	6.69E+02	9.94E+02	8.09E+02	1.12E+03	8.17E+02	1.67E+03	1.17E+03	7.78E+02
f_{24}	*Mean*	3.41E+02(+)	3.62E+02(+)	3.27E+02(+)	2.81E+02(+)	2.83E+02(+)	3.18E+02(+)	3.12E+02(+)	3.41E+02(+)	2.74E+02
	Std.	1.05E+01	6.23E+00	1.15E+01	1.34E+01	9.54E+00	1.14E+01	1.63E+01	1.14E+01	9.56E+00
f_{25}	*Mean*	3.86E+02(+)	4.12E+02(+)	3.84E+02(+)	3.57E+02(+)	3.23E+02 (=)	3.87E+02(+)	3.36E+02(=)	3.80E+02(+)	3.29E+02
	Std.	1.45E+01	7.32E+00	1.49E+01	1.22E+01	1.01E+01	8.28E+00	1.52E+01	9.91E+00	1.71E+01
f_{26}	*Mean*	3.80E+02(+)	2.02E+02(-)	3.42E+02(=)	2.00E+02 (-)	3.52E+02(+)	2.00E+02(-)	3.61E+02(+)	3.79E+02(+)	3.23E+02
	Std.	6.36E+01	3.60E-01	6.04E+01	6.46E-02	4.75E+01	8.77E-02	4.27E+01	5.59E+01	6.72E+01
f_{27}	*Mean*	1.61E+03(+)	1.52E+03(+)	1.22E+03(+)	1.22E+03(+)	1.17E+03(+)	1.40E+03(+)	1.29E+03(+)	1.53E+03(+)	1.01E+03
	Std.	8.59E+01	5.28E+02	9.62E+01	1.52E+02	8.71E+01	2.79E+02	9.00E+01	9.30E+01	1.27E+02
f_{28}	*Mean*	1.36E+03(+)	4.00E+02(=)	1.41E+03(+)	4.00E+02(=)	6.41E+02(+)	4.00E+02(=)	8.17E+02(+)	2.29E+03(+)	4.00E+02
	Std.	6.70E+02	9.69E-10	8.22E+02	2.02E-11	4.45E+02	1.56E-10	7.23E+02	8.90E+02	4.55E-13
(#)	+	24	20	22	16	15	17	16	27	
(#)	-	2	4	0	7	7	6	4	0	

multimodal functions on $50D$ case.

4.2.3. Composition functions (f_{21}-f_{28})

In the third set of experiments, 8 composition functions, which are very difficult to be optimized, are explored. The experimental results show that EPSO yields the most outstanding performance on the composition functions on $10D$ case since it achieves the best result on 3 out of the 8 functions, followed by CCPSO-ISM. Although MAPSO cannot yield the best performance on any composition function on $10D$ case, it offers the best solutions on 1 and 4 out of the 8 composition functions on $30D$ and $50D$ cases. Meanwhile, HCLPSO also shows very reliable performance on the higher dimensions cases. On the contrary, EPSO cannot display promising performance on $30D$ and $50D$ cases though it offers the best performance on $10D$. Thus, we can see that MAPSO and HCLPSO have a better scalability performance than other peer algorithms on the difficult composition functions.

4.2.4. Results of statistical comparisons

Statistical comparisons are also crucial for obtaining reliable conclusions of intelligence algorithms [58]. Thus, to investigate whether MAPSO is significantly better or worse than the other peer algorithms on CEC2013 test suite, a two-tailed t-test and a Friedman-test are carried out in this section.

1) t-test results. The results of t-test of the three dimension cases are presented in Table 2, 3, and 4, respectively, in which the symbols "(+)", "(-)" and "(=)" denote that MAPSO is significantly better than, significantly worse than, and almost the same as the corresponding competitor, respectively. Furthermore, the symbols "(#)+" and "(#)-" in the last two rows denote the number of "(+)" and "(-)", respectively.

It can be seen from Table 2 - Table 4 that MAPSO significantly outperforms other 8 existing PSO variants on majority of the tested functions on the three dimensionality cases. Although MAPSO achieves the same performance as H-CLPSO and a slightly better result than EPSO on $10D$ case functions, in terms of the t-test results, MAPSO offers significantly promising results than the two competitors on the higher dimensionality cases, i.e., $30D$ and $50D$ functions.

The t-test results manifest that MAPSO not only has favorable performance, in terms of solution accuracy, but also displays more reliable scalability than other competitors, especially on higher dimension cases.

2) Friedman-test results. To further illustrate the comprehensive performance of all the peer algorithms, and propose an overview of them, the Friedman-test is conducted in this part. The results of Friedman-test are listed in Table 5, in which the algorithms are listed in ascending order, in terms of the values of rankings (the lower the better). Furthermore, we also separately carry out Friedman-test of all peer algorithms on the three dimension cases (i.e., $10D$, $30D$, and $50D$), the results of which are also presented in Table 5.

Table 5: Friedman-test of all compared algorithms on CEC2013 test suite

Average	Over all		$10D$		$30D$		$50D$	
Rank	Algorithm	Ranking	Algorithm	Ranking	Algorithm	Ranking	Algorithm	Ranking
1	MAPSO	3.02	HCLPSO	3.11	MAPSO	2.91	MAPSO	2.61
2	HCLPSO	3.47	MAPSO	3.54	GLPSO	3.45	GLPSO	3.18
3	GLPSO	3.74	EPSO	3.57	HCLPSO	3.93	HCLPSO	3.38
4	EPSO	4.02	TAPSO	4.36	TAPSO	3.93	TAPSO	4.21
5	TAPSO	4.17	GLPSO	4.59	EPSO	3.93	EPSO	4.57
6	CCPSO-ISM	5.70	CCPSO-ISM	5.45	CCPSO-ISM	5.46	SRPSO	5.88
7	SRPSO	5.96	SRPSO	5.59	SRPSO	6.41	CCPSO-ISM	6.18
8	AWPSO	7.45	AWPSO	6.86	PSODDS	7.30	PSODDS	7.20
9	PSODDS	7.48	PSODDS	7.95	AWPSO	7.68	AWPSO	7.80

It can be observed from Table 5 that HCLPSO yields the most promising result on the $10D$ case, followed by MAPSO and EPSO. On the two higher dimensionality cases, i.e., $30D$ and $50D$, MAPSO displays the best performance, followed by GLPSO, HCLPSO, and TAPSO. The overall performance on the three dimensionality cases verifies that MAPSO offers the most favorable performance, while HCLPSO and GLPSO also display very outstanding characteristics.

4.2.5. Final rank of all peer algorithms

To evaluate overall performance of all the peer algorithms and give a final rank (FR) of them, an evaluation method based on a score of 100 is applied in this research. There are two criteria, in which higher weights are given for higher dimensions, involved in the evaluation method. The criteria and the entire evaluation are detailed in follows.

The one criterion is 50% summation of error (SE) values for all dimensions, which is defined as follows.

$$SE = 0.1 \times \sum_{i=1}^{28} f_{10D} + 0.3 \times \sum_{i=1}^{28} f_{30D} + 0.6 \times \sum_{i=1}^{28} f_{50D}$$

where f is the error values for all the functions and SE is the sum of errors, based on which the score for this part as follows.

$$Score1 = (1 - \frac{SE - SE_{min}}{SE}) \times 50$$

where SE_{min} is the minimal sum of errors from all the algorithms.

Another one criterion is 50% summation of rank (SR) results for each problem in each dimension, which is defined as follows.

$$SR = 0.1 \times \sum_{i=1}^{28} rank_{10D} + 0.3 \times \sum_{i=1}^{28} rank_{30D} + 0.6 \times \sum_{i=1}^{28} rank_{50D}$$

where SR is the sum of ranks, and then the score for this part as follows.

$$Score2 = (1 - \frac{SR - SR_{min}}{SE}) \times 50$$

where SR_{min} is the minimal sum of ranks from all the algorithms.

Then, we can combine the above two parts to find the final score of each algorithm as follows.

$$Score = Score1 + Score2$$

From the results of FR as well as $Score$ presented in Table 6 we can see that MAPSO offers the best overall performance measured by FR, while TAPSO is the second best, followed by GLPSO, HCLPSO and EPSO. Moreover, MAPSO not only attains the highest value of FR, but also performs best results on *Score1* and *Score2*.

4.3. Comparison on CEC2017 test suite

In this part, a more recent benchmark suite, i.e., CEC 2017 test suite, is selected to verity whether MAPSO can offer more reliable performance than the peer algorithms on the new test suite. Due to the space limitation, only one dimension case, i.e., D=30, is tested.

Table 6: Scores and final rank of all peer algorithms

Algorithms	*Score1*	*Score2*	*Score*	*FR*
PSODDS	0.21	18.83	19.04	8
CCPSO-ISM	0.48	23.34	23.82	6
SRPSO	0.37	22.87	23.25	7
HCLPSO	7.48	37.91	45.39	4
GLPSO	19.43	40.04	59.47	3
EPSO	3.84	31.24	35.08	5
TAPSO	36.54	34.13	70.67	2
AWPSO	0.04	18.10	18.14	9
MAPSO	50.00	50.00	100.00	1

4.3.1. Results of solution accuracy

In this part, the performance of solution accuracy is compared, in terms of mean value(*Mean*) and standard deviation (*Std.*), the results of which are presented in Table 7. From the table we can see that MAPSO, HCLPSO, and GLPSO display the same performance on the 3 unimodal functions (F_1-F_3) measured by the number of the best *Mean* values. However, MAPSO dominates other peer algorithms since it offers the best results on 4 out of the 10 simple multimodal functions (F_4-F_{10}). On the 20 complicated benchmark functions, i.e., 10 hybrid functions (F_{11}-F_{20}) and 10 composition functions (F_{21}-F_{30}), MAPSO exhibits more outstanding characteristics, followed by HCLPSO, EPSO, GLPSO, and TAPSO. Together with the results presented in Table 6, we can obtain that the 5 PSO variants offer more comprehensive performance than the other 4 peer algorithms on the two CEC test suites.

4.3.2. Results of time usage

In this part, an experiment is conducted to compare the time usages of the 9 algorithms, the results of which are listed in Table 8. From the results we obtain that EPSO is the most time consuming algorithm measured by the average time usage, followed by MAPSO, TAPSO, and GLPSO. On the contrary, AWPSO, PSODDS, and PSO-ISM consume less time on the test suite though they cannot display promising performance, in terms of the solutions accuracy.

Table 7: Comparison results of solution accuracy on CEC2017 test suite(D=30).

		PSODDS	CCPSO-ISM	SRPSO	HCLPSO	GLPSO	EPSO	TAPSO	AWPSO	MAPSO
F_1	*Mean*	4.75E+06(=)	5.46E+03(+)	9.42E+07(+)	7.49E+01 (-)	3.28E+03(=)	1.05E+02(-)	1.93E+03(=)	2.46E+09(+)	1.78E+03
	Std.Dev.	3.39E+07	2.08E+03	3.30E+08	1.48E+02	4.21E+03	1.50E+02	5.37E+03	2.67E+09	1.89E+03
F_2	*Mean*	7.69E+26(=)	7.62E+11(+)	5.00E+29(=)	5.23E+06(=)	1.52E+01(=)	3.00E+10(+)	6.52E+18(+)	3.12E+33(+)	4.71E+02
	Std.Dev.	5.49E+27	1.02E+12	3.56E+30	3.12E+07	2.57E+01	8.18E+10	4.65E+19	1.54E+34	2.83E+03
F_3	*Mean*	2.86E+02(+)	4.93E+04(+)	4.08E-04(=)	1.08E-03(=)	2.69E-06 (=)	3.54E+00(+)	3.40E-02(+)	1.00E+04(+)	1.14E-03
	Std.Dev.	7.90E+02	7.88E+03	7.83E-04	3.49E-03	1.28E-05	6.96E+00	1.05E-01	2.12E+04	2.98E-03
F_4	*Mean*	9.03E+01(=)	7.60E+01(-)	1.14E+02(+)	6.80E+01(-)	3.69E+01 (-)	5.51E+01(-)	1.14E+02(+)	2.81E+02(+)	9.10E+01
	Std.Dev.	2.25E+01	1.87E+01	7.57E+01	2.17E+01	3.07E+01	2.80E+01	4.68E+01	1.76E+02	2.20E+01
F_5	*Mean*	1.08E+02(+)	1.34E+02(+)	5.14E+01(+)	4.19E+01(+)	4.29E+01(+)	4.53E+01(+)	9.41E+01(+)	8.15E+01(+)	3.08E+01
	Std.Dev.	2.18E+01	1.98E+01	1.27E+01	8.28E+00	1.25E+01	6.89E+00	3.03E+01	2.75E+01	1.08E+01
F_6	*Mean*	1.14E+01(+)	1.33E-01(+)	5.41E-01(+)	3.46E-13 (=)	6.03E-05(+)	5.04E-13(=)	5.48E+00(+)	4.12E+00(+)	2.80E-06
	Std.Dev.	6.44E+00	2.38E-01	8.96E-01	1.33E-13	1.13E-04	1.51E-13	4.16E+00	3.05E+00	7.43E-06
F_7	*Mean*	1.34E+02(+)	1.04E+02(+)	7.51E+01(+)	8.15E+01(+)	7.56E+01(+)	8.75E+01(+)	1.70E+02(+)	1.12E+02(+)	5.30E+01
	Std.Dev.	2.53E+01	1.12E+01	1.53E+01	9.07E+00	1.08E+01	1.13E+01	4.80E+01	2.23E+01	1.01E+01
F_8	*Mean*	9.12E+01(+)	1.35E+02(+)	5.28E+01(+)	4.30E+01(+)	4.66E+01(+)	4.88E+01(+)	8.98E+01(+)	8.02E+01(+)	2.51E+01
	Std.Dev.	2.20E+01	1.73E+01	1.45E+01	8.73E+00	1.31E+01	8.07E+00	1.98E+01	2.33E+01	8.93E+00
F_9	*Mean*	1.36E+03(+)	5.77E+03(+)	5.18E+00(+)	8.08E+00(+)	3.27E+00(+)	6.28E+01(+)	5.85E+02(+)	3.52E+02(+)	5.68E-02
	Std.Dev.	7.98E+02	1.21E+03	1.14E+01	8.47E+00	4.21E+00	3.92E+01	4.44E+02	4.08E+02	1.36E-01
F_{10}	*Mean*	3.33E+03(+)	2.79E+03(=)	2.45E+03(-)	2.11E+03(-)	2.53E+03(=)	1.94E+03 (-)	2.76E+03(=)	2.85E+03(=)	2.68E+03
	Std.Dev.	5.55E+02	3.40E+02	5.84E+02	3.41E+02	5.52E+02	2.97E+02	5.28E+02	6.26E+02	5.88E+02
F_{11}	*Mean*	1.37E+02(+)	1.51E+02(+)	1.03E+02(+)	5.77E+01(+)	4.22E+01(+)	2.73E+01 (=)	1.08E+02(+)	2.00E+02(+)	2.93E+01
	Std.Dev.	5.75E+01	3.39E+01	4.56E+01	2.05E+01	2.75E+01	8.57E+00	4.03E+01	1.11E+02	2.36E+01
F_{12}	*Mean*	1.59E+06(+)	6.81E+05(+)	1.03E+06(+)	2.53E+04(+)	2.55E+04(+)	3.19E+05(+)	9.40E+05(+)	1.61E+08(+)	1.95E+04
	Std.Dev.	3.97E+06	3.31E+05	2.92E+06	1.36E+04	1.28E+04	1.72E+05	1.62E+06	3.26E+08	1.06E+04
F_{13}	*Mean*	1.12E+05(=)	1.97E+03(-)	1.59E+06(=)	4.03E+02 (-)	1.27E+04(=)	4.96E+02(-)	1.25E+04(-)	3.91E+07(+)	1.41E+04
	Std.Dev.	6.19E+05	1.22E+03	1.00E+07	2.83E+02	1.26E+04	7.18E+02	1.55E+04	1.93E+08	1.34E+04
F_{14}	*Mean*	1.13E+04(+)	3.11E+04(+)	1.09E+04(+)	3.87E+03(+)	3.49E+03(=)	1.65E+04(+)	2.44E+04(+)	1.79E+04(+)	2.37E+03
	Std.Dev.	2.47E+04	2.49E+04	2.70E+04	3.65E+03	4.48E+03	1.52E+04	5.58E+04	3.34E+04	2.45E+03
F_{15}	*Mean*	7.62E+03(=)	2.47E+02(-)	6.84E+03(=)	2.49E+02(-)	4.82E+03(-)	2.20E+02 (-)	6.75E+03(-)	4.20E+04(+)	8.63E+03
	Std.Dev.	8.56E+03	8.58E+01	7.33E+03	3.14E+02	5.75E+03	2.70E+02	8.56E+03	6.34E+04	1.06E+04
F_{16}	*Mean*	1.09E+03(+)	6.58E+02(+)	5.36E+02(+)	5.33E+02(+)	7.25E+02(+)	5.96E+02(+)	1.02E+03(+)	8.18E+02(+)	3.51E+02
	Std.Dev.	2.92E+02	1.34E+02	1.42E+02	1.57E+02	2.88E+02	1.73E+02	2.71E+02	2.88E+02	2.35E+02
F_{17}	*Mean*	5.47E+02(+)	2.41E+02(+)	2.34E+02(+)	1.06E+02 (=)	2.42E+02(+)	1.70E+02(+)	4.10E+02(+)	3.11E+02(+)	1.12E+02
	Std.Dev.	2.19E+02	8.46E+01	1.14E+02	7.65E+01	1.47E+02	8.70E+01	1.79E+02	1.58E+02	7.15E+01
F_{18}	*Mean*	8.87E+04(=)	1.34E+05(+)	1.69E+05(+)	9.46E+04(=)	6.89E+04 (=)	1.39E+05(+)	1.48E+05(+)	5.66E+05(+)	8.20E+04
	Std.Dev.	6.00E+04	5.90E+04	1.97E+05	5.07E+04	5.21E+04	7.47E+04	2.18E+05	1.02E+06	5.04E+04
F_{19}	*Mean*	1.85E+04(=)	8.13E+01 (-)	3.42E+04(=)	2.50E+02(-)	7.00E+03(=)	2.21E+02(-)	1.19E+04(+)	2.47E+05(+)	6.37E+03
	Std.Dev.	5.36E+04	3.16E+01	1.42E+05	3.39E+02	9.24E+03	3.09E+02	1.51E+04	7.76E+05	7.88E+03
F_{20}	*Mean*	5.32E+02(+)	3.40E+02(+)	2.05E+02(+)	1.36E+02(=)	2.47E+02(+)	2.18E+02(+)	4.26E+02(+)	2.99E+02(+)	1.13E+02
	Std.Dev.	2.08E+02	8.69E+01	5.92E+01	7.56E+01	1.42E+02	9.92E+01	1.69E+02	1.27E+02	6.54E+01
F_{21}	*Mean*	3.13E+02(+)	2.74E+02(+)	2.55E+02(+)	2.42E+02(+)	2.45E+02(+)	2.44E+02(+)	2.79E+02(+)	2.94E+02(+)	2.28E+02
	Std.Dev.	2.93E+01	1.02E+02	1.42E+01	2.22E+01	1.29E+01	3.00E+01	2.29E+01	1.75E+01	8.63E+00
F_{22}	*Mean*	1.27E+03(+)	8.00E+02(+)	4.82E+02(+)	1.00E+02 (=)	1.70E+02(+)	1.00E+02 (=)	1.01E+02(=)	2.12E+03(+)	1.00E+02
	Std.Dev.	1.75E+03	1.32E+03	8.79E+02	8.95E-01	4.97E+02	3.15E+00	2.81E+00	1.46E+03	7.08E-01
F_{23}	*Mean*	5.16E+02(+)	4.54E+02(+)	4.64E+02(+)	3.98E+02(+)	3.95E+02(+)	3.99E+02(+)	3.86E+02(=)	5.90E+02(+)	3.78E+02
	Std.Dev.	3.72E+01	4.91E+01	4.44E+01	1.11E+01	1.25E+01	1.07E+01	3.48E+01	7.49E+01	1.08E+01
F_{24}	*Mean*	5.80E+02(+)	5.53E+02(+)	5.65E+02(+)	4.55E+02(=)	4.69E+02(+)	4.99E+02(+)	4.42E+02 (-)	7.13E+02(+)	4.51E+02
	Std.Dev.	4.34E+01	1.98E+02	6.59E+01	7.56E+01	1.57E+01	4.57E+01	3.72E+01	1.18E+02	1.11E+01
F_{25}	*Mean*	3.98E+02(+)	3.89E+02(=)	3.93E+02(+)	3.87E+02(-)	3.89E+02(=)	3.86E+02 (-)	4.02E+02(=)	4.11E+02(=)	3.88E+02
	Std.Dev.	2.33E+01	1.58E+00	1.31E+01	9.88E-01	1.10E+01	1.55E+00	2.02E+01	7.28E+01	5.14E+00
F_{26}	*Mean*	2.20E+03(+)	5.76E+02(-)	1.34E+03(=)	3.23E+02 (-)	1.16E+03(=)	4.78E+02(-)	2.21E+03(+)	2.29E+03(+)	1.18E+03
	Std.Dev.	8.91E+02	4.05E+02	6.27E+02	2.76E+02	5.95E+02	5.21E+02	5.86E+02	7.62E+02	1.74E+02
F_{27}	*Mean*	5.57E+02(+)	5.18E+02(+)	5.48E+02(+)	5.13E+02(=)	5.18E+02(+)	5.13E+02(=)	5.10E+02 (=)	6.09E+02(+)	5.12E+02
	Std.Dev.	2.70E+01	4.01E+00	3.66E+01	7.55E+00	9.14E+00	4.20E+00	4.18E+00	9.38E+01	1.30E+01
F_{28}	*Mean*	4.53E+02(+)	4.51E+02(+)	4.35E+02(+)	3.50E+02(=)	3.31E+02 (-)	3.47E+02(-)	3.73E+02(=)	6.06E+02(+)	3.66E+02
	Std.Dev.	4.33E+01	1.19E+01	4.47E+01	5.31E+01	5.10E+01	4.28E+01	8.71E+01	1.60E+02	5.49E+01
F_{29}	*Mean*	9.87E+02(+)	7.36E+02(+)	6.34E+02(+)	5.10E+02(=)	6.12E+02(+)	5.18E+02(=)	9.51E+02(+)	5.19E+02(=)	4.97E+02
	Std.Dev.	2.37E+02	7.78E+01	1.08E+02	6.17E+01	1.60E+02	8.74E+01	2.13E+02	1.70E+02	5.45E+01
F_{30}	*Mean*	9.72E+04(+)	4.75E+03(=)	8.98E+03(+)	3.78E+03 (-)	4.85E+03(=)	4.13E+03(-)	2.12E+04(+)	5.15E+05(+)	5.27E+03
	Std.Dev.	3.25E+05	2.26E+03	1.29E+04	8.85E+02	2.31E+03	7.92E+02	1.86E+04	9.02E+05	2.12E+03
(#)	+	23	22	23	10	16	15	20	27	
(#)	-	0	5	1	9	3	10	3	0	

Thus, we can draw a preliminary conclusion that it is time consuming auxiliary operators in the outstanding PSO variants cause them to display more favorable performance. However, we also notice that differences of the time usages among the 9 peer algorithms on the complicated functions are less than that on the simple functions since the evaluation process, rather than the auxiliary operators, in the complicated functions is more time consuming. In fact, from an experiment we find out that the evaluation process accounts for about 25% of the entire time usage when optimizing F_1, while the proportion is about 80% when optimizing F_{30}. Thus, we regard that MAPSO is more suitable for complicated problems rather than simple unimodal problems.

Table 8: Comparison of time usage on CEC2017 test suite (in second)

	PSODDS	PSO-ISM	SRPSO	HCLPSO	GLPSO	EPSO	TAPSO	AWPSO	MAPSO
F_1	3.01	1.91	2.09	3.14	4.24	5.42	5.53	2.15	5.84
F_2	3.16	2.21	2.11	3.31	4.43	6.01	5.28	2.34	5.44
F_3	3.04	1.99	1.97	3.03	4.18	5.92	5.19	2.22	5.37
F_4	2.99	1.86	2.03	2.91	4.04	4.94	5.22	2.29	5.31
F_5	3.31	2.35	2.32	3.34	4.37	5.94	5.37	2.54	5.50
F_6	4.15	3.26	3.23	4.17	5.09	6.36	6.53	3.47	6.49
F_7	3.74	2.43	2.44	3.28	4.39	5.89	6.05	2.67	5.69
F_8	3.44	2.42	2.40	3.45	4.42	6.50	5.47	2.61	5.52
F_9	3.65	2.50	2.43	3.38	4.52	5.50	5.58	2.61	5.50
F_{10}	3.67	2.59	2.52	3.89	4.50	7.93	5.62	2.90	6.27
F_{11}	3.18	2.07	2.15	3.01	4.30	6.69	5.16	2.57	5.30
F_{12}	3.49	2.24	2.35	3.24	4.19	5.57	5.55	2.73	5.59
F_{13}	3.42	2.12	2.21	3.46	4.53	6.90	5.39	3.13	5.31
F_{14}	3.63	2.51	2.51	3.53	4.10	6.41	5.82	2.84	6.06
F_{15}	3.32	2.04	2.38	3.08	4.26	6.97	5.33	2.45	5.69
F_{16}	3.35	2.23	2.44	3.43	4.30	8.61	5.60	2.59	5.55
F_{17}	4.22	3.17	3.39	4.33	5.05	9.79	6.39	3.47	6.64
F_{18}	3.42	2.24	2.34	3.34	4.30	7.28	5.59	2.55	5.58
F_{19}	8.62	7.30	7.55	8.58	9.28	12.30	10.63	7.85	11.07
F_{20}	4.34	3.30	3.37	4.42	5.15	8.77	6.66	3.47	7.12
F_{21}	4.94	3.85	4.03	4.98	5.70	8.19	7.15	4.10	7.05
F_{22}	5.15	4.29	4.32	5.28	6.16	7.54	7.44	4.44	7.32
F_{23}	5.67	4.55	4.82	6.04	6.51	8.40	7.75	4.94	7.86
F_{24}	6.05	4.85	5.23	6.23	6.97	9.14	7.75	5.36	8.72
F_{25}	5.47	4.30	4.50	5.61	6.50	8.99	7.46	4.98	7.91
F_{26}	6.51	4.36	5.69	6.49	7.30	8.88	8.17	5.82	9.22
F_{27}	7.23	5.94	6.30	7.68	7.85	15.15	9.06	6.62	9.47
F_{28}	6.28	5.04	5.51	6.25	7.22	8.81	8.31	5.54	8.53
F_{29}	5.84	4.52	4.41	5.93	6.38	8.42	7.25	4.80	7.99
F_{30}	9.90	8.45	8.91	9.91	10.50	15.12	11.47	9.25	12.24
Avg.	4.61	3.46	3.59	4.62	5.49	7.94	6.66	3.84	6.91

5. Effectiveness of involved strategies

There are two newly introduced strategies in MAPSO, i.e., ALE and APS. To measure how the two strategies affect the performance of MAPSO, a series of experiments is carried out in this section. Due to the space limitation, only CEC2013 test suite with D=30 is tested in this work.

5.1. Effectiveness of ALE

In MAPSO, each particle in a swarm chooses its own learning exemplars to perform search process according to the particle's fitness. To verify effectiveness of ALE, three MAPSO variants, named as MAPSO$_E$, MAPSO$_I$, and MAPSO$_M$, are used to compare with MAPSO. Note that, MAPSO$_E$, MAPSO$_I$, and MAPSO$_M$ denote that all particles in the population select their exemplars from elites, inferiors, and mediocrities, respectively. The meanings of elites, inferiors, and mediocrities can refer to Figure 1. The experimental results measured by the solution accuracy and t-test results are presented in Table 9.

It can be observed from Table 9 that MAPSO yields the best results on 17 out of the 28 benchmark functions, in terms of the mean solution accuracy, followed by MAPSO$_M$, MAPSO$_I$ and MAPSO$_E$.

Among the 5 unimodal functions (f_1 - f_5), MAPSO$_E$ and MAPSO$_M$ display more favorable performance than MAPSO measured by the t-test results. The comparison results indicate that selecting outstanding particles as exemplars is beneficial for unimodal functions. Furthermore, MAPSO$_I$ achieves more promising result than MAPSO on f_3. The result manifests that adopting inferior particles as exemplars plays a positive performance on the non-separable unimodal function to some extent.

The comparison results on the 15 basic multimodal functions (f_6 - f_{20}) show that MAPSO$_I$ dominates MAPSO$_E$ and MAPSO$_M$. Concretely, MAPSO$_I$ obtains the best results on 4 out of the 15 multimodal functions, followed by MAPSO$_M$ and MAPSO$_E$. The result verifies that using elites as learning exemplars is unfavorable for multimodal functions since it may cause a population

Table 9: Performance of ALE

	MAPSO	MAPSO$_E$	MAPSO$_I$	MAPSO$_M$
f_1	**0.00E+00**±0.00E+00	**0.00E+00**±0.00E+00(=)	**0.00E+00**±0.00E+00(=)	**0.00E+00**±0.00E+00(=)
f_2	1.49E+06±9.77E+05	9.46E+05±8.75E+05(-)	1.29E+06±6.57E+05(=)	**8.24E+05**±4.62E+05(-)
f_3	1.46E+07±1.31E+07	2.96E+06±6.82E+06(-)	**8.76E+05**±2.17E+06(-)	1.84E+06±4.09E+06(-)
f_4	**8.14E+01**±8.64E+01	2.67E+02±1.47E+02(+)	9.87E+02±5.88E+02(+)	3.83E+02±1.23E+02(+)
f_5	6.91E-14±5.42E-14	8.25E-14±5.12E-14(=)	**4.97E-14**±5.82E-14(=)	6.68E-14±5.51E-14(=)
f_6	**1.69E+01**±1.55E+00	1.82E+01±3.83E+01(+)	1.89E+01±3.65E+00(+)	1.90E+01±2.33E+00(+)
f_7	1.87E+01±7.53E+00	8.95E-01±9.73E-01(-)	**2.75E-01**±3.07E-01(-)	5.11E-01±2.51E-01(-)
f_8	**2.10E+01**±4.00E-02	**2.10E+01**±3.35E-02(=)	**2.10E+01**±3.04E-02(=)	**2.10E+01**±4.91E-02(=)
f_9	1.22E+01±2.35E+00	1.58E+01±2.88E+00(+)	**4.20E+00**±2.10E+00(-)	5.28E+00±1.44E+00(-)
f_{10}	**1.10E-01**±4.61E-02	1.55E-01±7.85E-02(+)	4.62E-01±3.05E-02(+)	5.55E-01±2.82E-01(+)
f_{11}	1.22E+01±4.72E+00	1.17E+01±2.79E+00(=)	**1.16E+01**±5.27E+00(=)	1.18E+01±6.76E+00(=)
f_{12}	**4.43E+01**±9.23E+00	7.18E+01±1.39E+01(+)	7.15E+01±2.95E+01(+)	8.16E+01±3.31E+01(+)
f_{13}	9.37E+01±2.22E+01	4.66E+01±1.99E+01(-)	2.67E+02±3.38E+01(+)	**3.72E+01**±1.61E+01(-)
f_{14}	**1.71E+02**±8.86E+01	1.53E+03±1.35E+03(+)	5.01E+03±2.61E+03(+)	2.85E+03±2.51E+03(+)
f_{15}	**3.30E+03**±6.41E+02	3.96E+03±5.90E+02(+)	3.52E+03±2.19E+03(=)	3.68E+03±4.84E+03(=)
f_{16}	**1.33E+00**±3.87E-01	2.53E+00±2.99E-01(+)	2.54E+00±2.24E-01(+)	2.45E+00±2.02E-01(+)
f_{17}	**4.18E+01**±2.86E+00	1.05E+02±6.32E+01(+)	1.74E+02±1.45E+01(+)	1.60E+02±3.07E+01(+)
f_{18}	**5.06E+01**±1.39E+01	1.83E+02±9.17E+00(+)	1.83E+02±9.20E+00(+)	1.79E+02±9.87E+00(+)
f_{19}	3.15E+00±6.71E-01	2.93E+00±5.75E-01(=)	3.71E+00±1.70E+00(=)	**2.59E+00**±3.87E-01(-)
f_{20}	**9.97E+00**±5.43E-01	1.02E+01±7.68E-01(=)	1.07E+01±2.49E-01(+)	1.06E+01±3.02E-01(+)
f_{21}	**2.92E+02**±5.06E+01	2.93E+02±4.66E+01(=)	2.97E+02±4.45E+01(=)	2.94E+02±4.48E+01(=)
f_{22}	**2.19E+02**±6.47E+01	4.19E+02±6.46E+01(+)	1.45E+03±2.12E+03(+)	4.33E+02±3.86E+03(+)
f_{23}	3.44E+03±5.71E+02	**2.15E+03**±9.66E+02(-)	2.76E+03±1.03E+03(-)	3.41E+03±5.88E+02(=)
f_{24}	2.26E+02±8.91E+00	**2.03E+02**±7.00E+00(-)	2.10E+02±1.52E-01(-)	2.21E+02±6.32E+00(=)
f_{25}	**2.60E+02**±8.91E+00	2.75E+02±1.08E+01(+)	2.76E+02±7.00E+00(+)	2.72E+02±7.26E+00(+)
f_{26}	**2.00E+02**±3.63E-02	2.06E+02±2.53E+01(+)	2.13E+02±3.42E+01(+)	**2.00E+02**±1.30E-02(=)
f_{27}	5.33E+02±8.32E+01	3.54E+02±7.89E+00(-)	3.12E+02±3.72E+00(-)	**3.03E+02**±2.18E+00(-)
f_{28}	**3.00E+02**±3.13E-13	**3.00E+02**±1.36E-13(=)	**3.00E+02**±0.00E+00(=)	**3.00E+02**±0.00E+00(=)

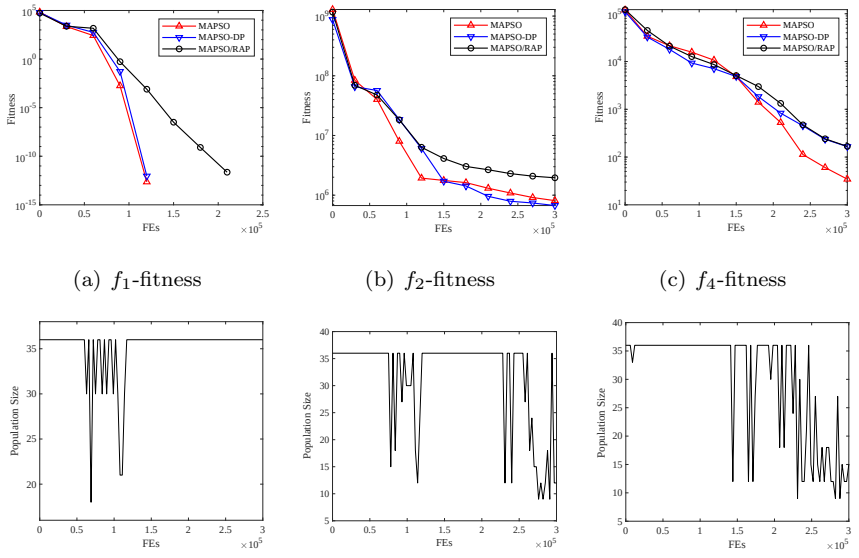
rapidly lost its diversity. In addition, MAPSO yields the most promising performance on the 15 multimodal functions since MAPSO achieves the best results on 10 functions, which significantly outperforms other MAPSO variants on most of the functions. The results verify the outstanding and comprehensive performance of ALE strategy.

It can be seen from the results on the 8 composition functions (f_{21} - f_{28}) that MAPSO surpasses other 3 peer algorithms measured by the number of the best results. Concretely, MAPSO yields the best performance on 5 out of the 8 composition functions, followed by MAPSO$_E$ and MAPSO$_M$.

From the comparison aforementioned, we can observe that the ALE strategy is beneficial for improving comprehensive performance of MAPSO on different types of functions.

5.2. *Effectiveness of APS*

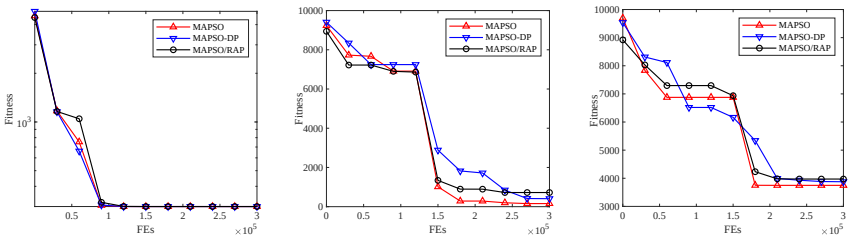
In MAPSO, APS strategy consisting an adding particles process and a deleting particles process is used to rational allocate the computational resource. In this section, effectiveness of the strategy is analyzed based on comparison results between MAPSO and other two MAPSO variants, i.e., MAPSO-DP and MAPSO/RAP. Note that, MAPSO-DP denotes that the **d**eleting **p**articles process is removed from MAPSO, while MAPSO/RAP indicates the adding individuals process in MAPSO is replaced by a **r**andom **a**dding **p**articles process. Due to the limitation of space, only 3 unimodal functions (f_1, f_2, and f_3) and 3 composition functions (f_{21}, f_{22}, and f_{23}) are selected as test functions in this section. The comparison results of the convergence process among MAPSO, MAPSO-DP and MAPSO/RAP on the unimodal functions and composition functions are illustrated in Figure 3 and Figure 4, respectively.



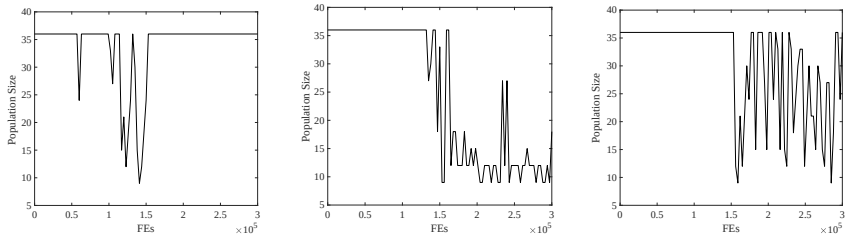
(a) f_1-fitness (b) f_2-fitness (c) f_4-fitness

(d) f_1-population size (e) f_2-population size (f) f_4-population size

Figure 3: Change trends of fitness and population size on 3 unimodal functions.



(a) f_{21}-fitness (b) f_{22}-fitness (c) f_{23}-fitness



(d) f_{21}-population size (e) f_{22}-population size (f) f_{23}-population size

Figure 4: Change trends of fitness and population size on 3 composition functions.

5.2.1. Performance of deleting particle process

In this part, we explain the performance of deleting particle process through the fitness change trends between MAPSO and MAPSO-DP as well as the population size change trends of MAPSO.

From Figure 3(a) we can observe that MAPSO, MAPSO-DP, and MAPSO/RAP find out the global optimal solution on the simple function f_1. However, MAPSO displays the highest convergence speed. The experimental result demonstrated in Figure 3(d) shows that the population size of MAPSO decreases during the early evolutionary process. Comparing the results illustrated in Figure 3(a) and Figure 3(d) we can see that it is the deleting particle process that enables MAPSO to has a higher accuracy speed than MAPSO-DP. For the other non-separable unimodal functions, i.e., f_2 and f_4, MAPSO also attains very promising performance. Specifically, from Figure 3(e) and Figure 3(f) we

can observe that MAPSO performs the deleting particle process mainly in the later evolutionary stage on f_2 and f_4. The phenomenon indicates that many smooth local irregularities in the two non-separable unimodal functions (i.e., f_2 and f_4) make MAPSO difficult to find out a global (or local) optimum during the early optimization process. When the population of MAPSO is located around to an optimum at the later evolutionary stage, MAPSO performs the deleting particle process, which causes MAPSO to offers the higher convergence speed on f_2 and f_4 at the later stage.

From the comparison results demonstrated in Figure 4 we can see that MAPSO dominates MAPSO-DP on 2 out of the 3 composition functions. Specially, MAPSO offers a faster convergence speed than MAPSO-DP on f_{22} and f_{23} at the later evolutionary stage. From Figure 4(e) and Figure 4(f) we can observe that the deleting particle process is not implemented at the initial evolutionary stage. At the later stage, while the population converges to optimal solutions, the deleting particle process is performed, resulting in the higher convergence speed at the stage.

5.2.2. Performance of adding particle process

In this part, the performance of the adding particle process is analyzed based on changing trends of fitness between MAPSO and MAPSO/RAP.

The experimental results demonstrated in Figure 3 indicate that MAPSO dominates MAPSO/RAP on the three simple unimodal functions. Specifically, MAPSO yields more accurate solutions and higher convergence speeds than MAPSO/RAP on the two non-separable unimodal functions (i.e., f_2 and f_4). Although MAPSO and MAPSO/RAP can find out the global optimum of the simple unimodal function f_1, MAPSO displays a higher convergence speed than MAPSO/RAP. The favorable performance of MAPSO on the 3 unimodal functions manifests that adding particles that generated by common genetic operators based on the elite archive A_E yields more favorable characteristics, in terms of solution accuracy and convergence speed, than adding randomly generated particles.

From Figure 4 we can see that MAPSO slightly outperforms MAPSO/RAP on the 3 complicated functions. Although both MAPSO and MAPSO/RAP cannot display very outstanding performance measured by solution accuracy, MAPSO yields relatively favorable characteristics than MAPSO/RAP.

The experimental results indicate that adding particle based on ordinary genetic operator allows MAPSO to attain competitive results on different types of functions, though the performance of it needs to be further improved on complicated functions.

5.2.3. Change of diversity

Furthermore, we also carry out a series of experiments to investigate how the three strategies affect the convergence speed and diversity of the population. In this study, the population deviation is denoted as (8).

$$diversity(ps) = \frac{1}{|ps| \cdot |L|} \cdot \sum_{i=1}^{|ps|} \sqrt{\sum_{j=1}^{D} (x_{ij} - \overline{x}_j)^2} \tag{8}$$

where $|ps|$ is the population size, $|L|$ is the length of the longest diagonal in the searching space, D is the dimensionality of the problem, x_{ij} is the j^{th} value of the i^{th} particle and $\overline{x}_j$ is the j^{th} value of the average point $\overline{x}$.

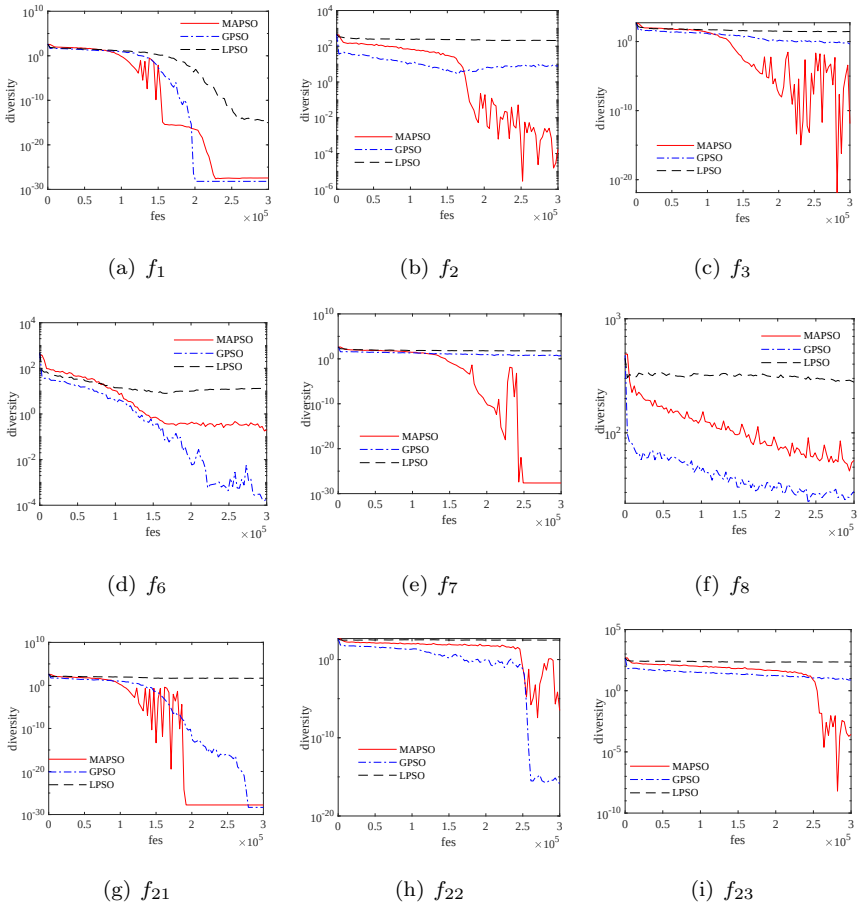
In the experiment, the global version PSO (GPSO) and the local version PSO (LPSO) are selected as peer algorithms. Due to the space limitation, only comparison results on 3 unimodal functions (f_1 - f_3), 3 basic multimodal functions (f_6 - f_8), and 3 composition functions (f_{21} - f_{23}) are demonstrated in Figure 5.

From the change trend in the simple unimodal functions *Sphere* (Figure 5(a)) we observe that population diversity of MAPSO declines at a higher rate in the former generations caused by the deleting particles process in the simple function, on the contrary, GPSO offers the highest decent speed in the latter generations since all particles have converged to the optimal solution. For Figure 5(b) and Figure 5(c) we see that MAPSO has the highest convergence speed in the latter generation on the 2 rotated unimodal functions, and the population diversity is well maintained in MAPSO in the former generations. Meanwhile,

fluctuations of the population diversity in the latter generations verify that the adding particles process in APS is beneficial for increasing the diversity, and then to enhance the exploration ability of MAPSO. The comparison results on the 3 rotated multimodal functions (see Figure 5(d) - Figure 5(f)) show that GPSO is more easily trapped into local optimal solutions than MAPSO and LPSO because GPSO has the highest decent speed of diversity during the entire evolution process. Although GPSO and LPSO cannot convergence in F_7, MAPSO offer a very fast convergence speed in the latter evolutionary process. The phenomenon verifies that it is those new added elite particles generated by DE-based breeding process that attract the population to (local) optimal solutions. On the three complicated functions (i.e., f_{21} -f_{22}) that all the competitors cannot offer promising performance, MAPSO also offers a higher convergence speed in the latter evolutionary process as well as maintain a promising population diversity in the former process. Together with the experimental results demonstrated in Figure 4(d) - Figure 4(f) we can obtain a preliminary conclusion that the APS not only beneficial for increasing the population diversity but also help for rationally utilizing computational resource.

6. Conclusion

In this paper, a multiple adaptive strategies based particle swarm optimization (MAPSO) in proposed, in which an entire population is divided into many small-sized sub-swarms. Based on the multiple sub-swarms mechanism, two adaptive strategies, i.e., adaptive of learning exemplars and adaptive of population size, are introduced to improve the comprehensive performance of MAPSO. Relying on the adaptive of learning exemplars strategy, which inspired by a common phenomenon that different persons can adaptive select their own exemplars, each particle in a sub-swarm can adaptive select its own learning exemplar according to its fitness value. Furthermore, the adaptive of population size is used to rationally distribute computational resources during the entire optimization process.



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

(d) f_6 (e) f_7 (f) f_8

(g) f_{21} (h) f_{22} (i) f_{23}

Figure 5: Change trends of diversity on 9 test functions.

To testify the performance of MAPSO, extensive experiments have been conducted to compare it with other 8 widely accepted PSO variants on CEC2013 and CEC2017 test suites. The experimental results have verified the promising performance of MAPSO. In addition, the advantages of the newly introduced strategies are experimentally verified by a set of experiments. From these comparison results, some conclusions can be drawn. First, the adaptive learning exemplars strategy can help a particle to choose proper learning exemplars, and then conduct a promising search behavior. Second, the deleting particle

process in the adaptive population size strategy can speed up the convergence of MAPSO since some unnecessary particles are removed from the population when current fitness landscape is easy to be optimized. Last, the adding particle process in the adaptive population size strategy can inject more helpful information generated by a DE algorithm into the current population.

Although the multiple adaptive strategies introduced in MAPSO attain competitive performance verified by the extensive experiments, we also find out that MAPSO is a relatively time consuming PSO variant especially on simple unimodal functions. Thus, we believe that MAPSO is more suitable for complicated problems rather than simple unimodal problems. Furthermore, there are some open issues need to be further studied. One issue is when and how to change learning exemplars for a specific particle. Another one issue is how to design a more efficient adaptive strategy to adjust the population size according to search behaviors of the population. Furthermore, when adding new particles into the current population, it is not negligible that generating promising particles with different characteristics to satisfy distinct requirements of evolutionary stages. Our future work will focus on the above problems, and enable the population to carry out high and effective search behaviors.

Acknowledgment. This study was funded by the National Natural Science Foundation of China (Grant No.: 61876136, 61806204, 61762036, 61663009, 61702239), the Natural Science Foundation of Jiangxi Province (Grant No.: 20171BAB202012), and the Research Project of Jiangxi Provincial Department of Communication and Transportation (No. 2017D0038).

Reference

[1] J.D. Ser, E. Osaba, D. Molina, *et al.*, Bio-inspired computation: Where we stand and what's next, Swarm Evol. Compt. 48(2019) 220-250.

[2] A. Rajasekhar, N. Lynn, S. Das, P.N. Suganthan, Computing with the collective intelligence of honey bees C A survey, Swarm Evol. Compt. 32(2017) 25-48.

[3] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proc. of International Symposium on Micro Machine and Human Science, ISMMHS'95, Nagoya, Japan, 1995, pp. 39-43.

[4] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proc. of IEEE International Conference on Neural Network, CNN'95, Perth, Australia, 1995, pp. 1942-1948.

[5] G. Ciuprina, D. Ioan, I. Munteanu, Use of intelligent-particle swarm optimization in electormagnetics, IEEE Trans. Magn. 38(2)(2002) 1037-1040.

[6] S.H. Ling, H.H.C. Iu, K, Y. Chan, H.K. Lam, B.C.W. Yeung, F.H. Leung, Hybrid particle swarm optimization with wavelet mutation and its industrial applications, IEEE Trans. Syst., Man, Cybern. B, Cybern. 34(2)(2004) 997-1006.

[7] H. Soh, Y.S. Ong, Q.C. Nguyen, Q.H. Nguyen, M.S. Habibullah, T. Hung, J.L. Kuo, Discovering unique, low-energy pure water isomers: Memetic exploration, optimization and landscape analysis, IEEE Trans. Evol. Comput. 14(3)(2010) 419-437.

[8] V. Kadirkamanathan, K. Selvarajah, P.J. Fleming, Stability analysis of the particle dynamics in particle swarm optimizer, IEEE Trans. Evol. Comput. 10(3)(2006) 245-255.

[9] J. Kennedy, Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance, in: Proc. of IEEE Congress on Evolutionary Computation, CEC'99, Washington DC, USA, Jul. 1999, pp. 1931-1938.

[10] A. Ratnaweera, S.K. Halgamuge, and H.C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Trans. Evol. Comput. 8(3)(2004) 240-255.

[11] J. Kennedy, R. Mendes, Population structure and particle swarm performance, in: Proc. of IEEE Congress on Evolutionary Computation, CEC'02, Honolulu, HI, May, 2002, pp. 1671-1676.

[12] R. Tanabe, A.S. Fukunaga, Improving the search performance of SHADE using linear population size reduction, in: Proc. of IEEE Congress on Evolutionary Computation, CEC'14, Beijing, China, 2014, pp. 1658-1665.

[13] W. Zhu, Y. Tang, J. Fang, W. Zhang, Adaptive population tuning scheme for differential evolution. Inf. Sci. 223(2013) 164-191.

[14] A.E. Eiben, M.C. Schut, A.R. de Wilde, Is self-adaptation of selection pressure and population size possible - A case study, in: Proc. the 9th International Conference on Parallel Problem Solving From Nature, PPSN'06, Reykjavik, 2006, pp. 900-909.

[15] N. Hansen, S.D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es), Evol. Comput., 11(1)(2003) 1-18.

[16] N.R. Samal, A. Konar, S. Das, A. Abraham, A Closed Loop Stability Analysis and Parameter Selection of the Particle Swarm Optimization Dynamics for Faster Convergence, in: Proc. of IEEE Congress on Evolutionary Computation, CEC'07, Singapore, 2007, pp. 1769-1776.

[17] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proc. of IEEE World Congress on Computational Intelligence, Anchorage, AK, May, 1998, pp. 68-73.

[18] K.R. Harrison, A.P. Engelbrecht, B.M. Omubuki-Berman, Self-adaptive particle swarm optimization: a review and analysis of convergence, Swarm Intell. 12(2018) 187-226.

[19] Z.H. Zhan, J. Zhang, Y. Li, H.S. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst., Man, Cybern. B, Cybern. 39(6)(2009) 1362-1381.

[20] W.B. Liu, Z.D. Wang, Y. Yuan, N.Y. Zeng, K. Hone, X.H. Liu, A novel sigmoid-function-based adaptive weighted particle swarm optimizer, IEEE Trans. Cybern. doi: 10.1109/TCYB.2019.2925015

[21] X.W. Xia, Y. Xing, B. Wei, Y.L. Zhang, X. Li, X.L. Deng, L. Gui, A fitness-based multi-rule particle swarm optimization, Swarm Evol. Compt. 44(2019) 349-364.

[22] A. Hashemi, M. Meybodi, A note on the learning automata based algorithms for adaptive parameter selection in PSO. Appl. Soft Comput. 11(1)(2011) 689-705.

[23] X.W. Xia, L. Gui, G.L. He, B. Wei, Y.L. Zhang, F. Yu, H.R. Wu, Z.H. Zhan, An expanded particle swarm optimization based on multi-exemplar and forgetting ability, Inf. Sci. 508(2020) 105-120.

[24] Y. Juang, S. Tung, H. Chiu, Adaptive fuzzy particle swarm optimization for global optimization of multimodal functions, Inf. Sci. 181(20), 2011, 4539-4549.

[25] M.R. Tanweer, S. Suresh, N. Sundararajan, Self regulating particle swarm optimization algorithm, Inf. Sci. 294(C)(2015) 182-202.

[26] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler,maybe better, IEEE Trans. Evol. Comput. 8(3)(2004) 204-210.

[27] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baska, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10(3)(2006) 281-295.

[28] Z.H. Zhan, Y. Li, Y.H. Shi, Orthogonal learning particle swarm optimization, IEEE Trans. Evol. Comput. 15(6)(2011) 832-847.

[29] D.S. Wang, D.P. Tan, L. Liu, Particle swarm optimization algorithm: an overview, Soft Comput. 2018(22) 387-408.

[30] X. Jin, Y.Q. Liang, D.P. Tian, F.Z. Zhuang, Particle swarm optimization using dimension selection methods, Appl. Math. Comput. 219(10)(2013) 5185-5197.

[31] J.J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer, in: Proc. of IEEE Swarm Intelligence Symposium, SIS'05, Pasadena, CA, USA, 2005, pp. 124-129.

[32] S.Z. Zhao, P.N. Suganthan, S. Das, Dynamic multi-swarm particle swarm optimizer with subregional harmony search, in: Proc. of IEEE Congress on Evolutionary Computation, CEC'10, Barcelona, Spain, 2010, pp. 1983-1990.

[33] J. Liu, D. Ma, T.B. Ma, W. Zhang, Ecosystem particle swarm optimization, Soft Comput. 21(3)(2017) 1667-1691.

[34] N. Lynn, P.N. Suganthan, Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation, Swarm Evol. Comput. 24(2015), 11-24.

[35] M. R. Bonyadi, Z. Michalewicz, Particle swarm optimization for single objective continuous space problems: a review, Evol. Comput. 25(1)(2017) 1-54.

[36] P.J. Angeline, Using selection to improve particle swarm optimization, in: Proc. of IEEE Congress on Evolutionary Computation, CEC'98, Alaska, USA, May, 1998, pp. 84-89.

[37] Y.J. Gong, J.J. Li, Y.C. Zhou, Y. Li, H.S.H. Chung, Y.H. Shi, J. Zhang, Genetic learning particle swarm optimization, IEEE Trans. Cybern. 46(10)(2016), 2277-2290.

[38] X.W. Xia, L. Gui, F. Yu, H.R. Wu, B. Wei, Y.L. Zhang, Z.H. Zhan, Triple archives particle swarm optimization, IEEE Trans. Cybern. doi: 10.1109/TCYB.2019.2943928

[39] B.Y. Qu, J.J. Liang, P.N. Suganthan, Niching particle swarm optimization with local search for multi-modal optimization, Inf. Sci. 197(2012) 131-143.

[40] Y.J. Wang, Improving particle swarm optimization performance with local search for high-dimensional function optimization, Optim. Methods Softw. 25(5)(2010) 781-795.

[41] X.W. Xia, J.N. Liu, Z.B. Hu, An improved particle swarm optimizer based on tabu detecting and local learning strategy in a shrunk search space, Appl. Soft Comput. 23(5)(2014) 76-90.

[42] N. Lynn, P.N. Suganthan, Ensemble particle swarm optimizer, Appl. Soft Comput. 55(2017), 533-548.

[43] B. Xin, J. Chen, Z.H. Peng, An adaptive hybrid optimizer based on particle swarm and differential evolution for global optimization, Sci. China (Inform. Sci.), 53(5)(2010) 980-989.

[44] B. Xin, J. Chen, J. Zhang, H. Fang, Z.H. Peng, Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: A review and taxonomy, IEEE Trans. Syst. Man, Cybern. C, Appl. Rev. 42(5)(2012) 744-767.

[45] M.S. Kiran, M. Gndz, A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems, Appl. Soft Comput. 13(4)(2013) 2188-2203.

[46] K.B. Kela, L.D. Arya, Reliability optimization of radial distribution systems employing differential evolution and bare bones particle swarm optimization, J. Inst. Eng. India Ser. B. 95(3)(2014) 231-239.

[47] S. Sayah, A. Hamouda, A hybrid differential evolution algorithm based on particle swarm optimization for nonconvex economic dispatch problems, Appl. Soft Comput. 13(4)(2013) 1608-1619.

[48] A. Marco, d.O. Montes, S. Thomas, B. Mauro, D. Marco, Frankenstein's PSO: A composite particle swarm optimization algorithm, IEEE Trans. Evol. Comput. 13(5)(2009) 1120-1132.

[49] M.A. de Oca, T. Stützle, V.D.E. Ken, M. Dorigo, Incremental social learning in particle swarms, IEEE Trans. Syst., Man, Cybern. B, Cybern. 42(2)(2011) 368-384.

[50] T. Peram, K. Veeramachaneni, C.K. Mohan, Fitness-distance-ratio based particle swarm optimization, in: Proc. of IEEE Swarm Intelligence Symposium, SIS'03, Indianapolis, IN, Apr. 2003, pp. 174-181.

[51] G.H. Wu, R. Mallipeddi, P.N. Suganthan, Differential evolution with multi population based ensemble of mutation strategies, Inf. Sci. 329(C)(2016) 329-345.

[52] R. Storn, K. Price, Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11(4)(1997) 341-359.

[53] S. Das, S.S. Mullick, P.N. Suganthan, Recent advances in differential evolution-an updated survey, Swarm Evol. Comput. 27(2016) 1-30.

[54] J. Zhang, A. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13(5)(2009) 945-958.

[55] Y. Li, Z.H. Zhan, S. Lin, J. Zhang, X. Luo, Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems, Inf. Sci. 293(3)(2015) 370-382.

[56] J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization, Nanyang Technological Univ., Singapore, Tech. Rep., 2013.

[57] N.H. Awad, M.Z. Ali, J.J. Liang, B.Y.Qu, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and

Competition on Single Objective Real-Parameter Numerical Optimization, Nanyang Technological Univ., Singapore, Tech. Rep., 2016.

[58] J. Carrasco, S. García, M.M. Rueda, S. Das, F. Herrera, Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review, Swarm and Evolutionary Computation, 54(2020) 100665