Information Sciences 586 (2022) 176–191

Contents lists available at ScienceDirect

Information Sciences

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

PSO-sono: A novel PSO variant for single-objective numerical optimization

Zhenyu Meng [a,b,*], Yuxin Zhong [a], Guojun Mao [b], Yan Liang [c]

[a] Institute of Artificial Intelligence, Fujian University of Technology, Fuzhou, China
[b] Fujian Provincial Key Laboratory of Big Data Mining and Applications, Fujian University of Technology, Fuzhou, China
[c] School of Resources and Environment, University of Electronic Science and Technology of China, Chengdu, China

ARTICLE INFO

Article history:
Received 10 February 2021
Received in revised form 25 November 2021
Accepted 26 November 2021
Available online 4 December 2021

Keywords:
Adaptation scheme
Numerical optimization
Particle swarm optimization
Single-objective optimization

ABSTRACT

Particle Swarm Optimization(PSO) is a well-known and powerful meta-heuristic algorithm in Swarm Intelligence (SI), and it was invented by simulating the foraging behavior of bird flock in 1995. Recently, many different PSO variants were proposed to tackle different optimization applications, however, the overall performance of these variants were not satisfactory. In this paper, a new PSO variant is advanced to tackle single-objective numerical optimization, and there are three contributions mentioned in the paper: First, a sorted particle swarm with hybrid paradigms is proposed to improve the optimization performance; Second, novel adaptation schemes both for the ratio of each paradigm and the constriction coefficients are proposed during the iteration; Third, a fully-informed search scheme based on the global optimum in each generation is proposed which helps the algorithm to jump out the local optimum and improve the overall performance. A large test suite containing benchmarks from CEC2013, CEC2014 and CEC2017 test suites on real-parameter single-objective optimization is employed in the algorithm validation, and the experiment results show the competitiveness of our algorithm with the famous or recently proposed state-of-the-art PSO variants.

1. Introduction

Particle Swarm Optimization (PSO) algorithm is a meta-heuristic global optimization algorithm proposed by Kennedy and Eberhart [1] in 1995. The underlying physical model upon which the transition rules are based is one of emergent collective behavior arising out of social interaction of flocks of birds or schools of fish [2–4]. In PSO, each individual in the swarm is called a particle, which represents a potential solution with two main characteristics (vectors), namely the position and the velocity [5–8]. In the initialization stage, the position of the i^{th} particle (in the 0^{th} generation) can be initialized according to the following equation:

$$X_{i,0} = X_{\min} + rand_{1\times D}(0,1)\cdot(X_{\max} - X_{\min}) \tag{1}$$

* Corresponding author at: Institute of Artificial Intelligence, Fujian University of Technology, Fuzhou, China.
E-mail address: mzy1314@gmail.com (Z. Meng).

https://doi.org/10.1016/j.ins.2021.11.076
0020-0255/

where $X_{\min}$ and $X_{\max}$ are the bounds of the solution space [9–11]. There are two different approaches to the initialization of the velocities in PSO: one is that the velocities are initialized to random values within the velocity bounds [11–13] and the other is that the velocities are initialized to zeros [14–16]. For a single-objective optimization problem that is defined below:

$$\Omega^* \equiv \arg\min_{X\in\Omega} f(X) = \{X^* \in \Omega : f(X^*) \leqslant f(X), \forall X \in \Omega\} \tag{2}$$

where X denotes a D-dimensional vector in solution space Ω, and Ω^* denotes the set of best candidates of the solution space, the PSO algorithm may return tolerable solution after finding it or arriving at the maximum number of function evaluations.

The iteration paradigm of the canonical PSO is simple, and the velocity of the particle is only affected by its current velocity, its historical best location and the population's global best location, in other words, the particle only can learn from its own memory $X_{pbest,G}$ and the best of the population $X_{gbest,G}$. The insufficient use of population information showed many disadvantages not only in scientific research [17–24] but also in some engineering applications of PSO [25–30]. Therefore, Suganthan [17] proposed a dynamic neighborhood operator to enhance the PSO algorithm. The neighborhood in this algorithm was initialized with some particles at the beginning of the iteration and then gradually increased to the whole population when arriving at the end of the iteration. Mendes et al. [19] proposed a fully informed particle swarm optimizer in which all neighbors of a certain particle are considered as a source of influence, and then the size of the neighborhood determined the diversity of the influence. Liang et al. [12], in CLPSO algorithm, proposed a comprehensive learning strategy in which a particle's velocity is updated by other particles' historical best. This technique improved the performance of PSO on multi-modal objectives at the expense of slowing down the convergence speed. Nasir et al. [13] further extended the CLPSO algorithm by delimiting the exemplar particle to a dynamic neighborhood, and this further improved the diversity of particles, however, both the CLPSO and DNLPSO are time consuming. Lynn et al. [31] presented a review of population topology or sociometry that enhanced the population diversity on the performance improvement of PSO. By incorporating the neighborhood topology, these PSO variants obtained a better use of the population information and consequently obtained significant performance improvement regarding different objectives.

Besides exploiting the information within the population, PSO researchers also proposed parameter-based techniques for performance improvements [32–38]. Shi and Eberhart [32] found that the convergence speed was usually accelerated by adding an inertia weight in front of the velocity of PSO algorithm, therefore, the inertia weight incorporated iteration equation was employed in the following PSO variants. The authors also found that better optimization performance was obtained when employing a linear reduction strategy of the inertia weight [33], and the recommended inertia weight at the initialization was 0.9 and at the termination was 0.4 at the end of the iteration. Taherkhani and Safabakhsh [37] proposed a stability-based adaptive inertia weight PSO in which the inertia weight of each dimension of a particle was different, and these values were determined by the performance of a certain particle and the distance from its historical best position. Bratton and Kennedy [34] reviewed some advances of PSO algorithm and defined a standard PSO algorithm in which the iteration equation employed a constriction version and the inertia weight of the velocity is set to a constant value 0.72984. Bansal et al. [36] made a comparison of different PSO variants under different inertia weight strategies, and they found that the chaotic inertia weight strategy outperformed others from the optimization accuracy perspective while random inertia weight strategy had a better convergence speed. However, the comparison was conducted under a test suite containing a small number of benchmarks, and maybe this comparison was biased according to the "No Free Lunch Theorem" [39]. Harrison et al. [38] further examined 18 inertia weight strategies under a large test suite containing 60 benchmarks, and the results showed that employing a fixed constant inertia weight was also the good choice for an arbitrary optimization problem.

To summarize, there are three aspects that can improve the performance of PSO algorithm when tackling single-objective numerical optimization. The first can be the improvement of the iteration paradigm of all particles. This may involve the improvement of a single iteration paradigm or the improvement of a hybrid paradigms. The second can be the improvement of adaptation strategies either for the constriction coefficients or for the adaptive selection of an iteration paradigm. The third can be some fully-informed techniques which can improve the exploration capacity of a PSO variant or help it jump out some local optima. Here in this paper, we propose a novel PSO variant with hybrid paradigms and parameter adaptation schemes for single-objective numerical optimization, and the main highlights of our algorithm are given as follows:

1. A novel sorted particle swarm with hybrid paradigms is proposed in our algorithm which achieved a balance of exploitation and exploration.
2. Novel adaptation schemes both for the ratio of each sub-population and for the constriction coefficients are proposed during the iteration, which can enhance the advantage of each iteration paradigm.
3. A novel fully-informed search based on the global optimum in each generation is proposed which helps the algorithm jump out the local optimum and improve the overall performance.
4. A large test suite containing all the benchmarks from CEC2013, CEC2014 and CEC2017 test suites for real-parameter single-objective optimization is used for algorithm validation, which, to some extent, avoids the over-fitting problem in comparison with employing just one test suite containing a small number of benchmarks.

The rest of this paper is organized as follows: Section 2 presents the brief review of several PSO variants which are closely related to the proposed PSO variant in this paper. Section 3 gives the details of our algorithm, and the novel algorithm is verified in Section 4 under our test suite containing 88 benchmarks. Finally, the conclusion is given in Section 5.

2. Several closely related PSO variants

In this section, several closely related PSO variants are reviewed including the inertia weight PSO (iwPSO) [32], the Comprehensive Learning PSO (CLPSO) [12], the Dynamic Neighborhood Learning PSO (DNLPSO) [13], the Social Learning PSO (SLPSO) [40], the Ensemble PSO (EPSO) [41], and the modified PSO with chaotic-based inertia weight (MPSO) [42].

2.1. The iwPSO algorithm

In the PSO algorithms, a population containing PS particles is used for the tackling of optimization problems, and the population can be denoted as $\mathbf{P} = [X_{1,G}, X_{2,G}, X_{3,G}, \ldots, X_{i,G}, X_{PS,G}]$, where $X_{i,G}$ is a D-dimensional vector which denotes the i^{th} particle of the population in the G^{th} generation. All the algorithms in PSO branch search the solution space by particles' movements. The moving direction and step size of the i^{th} particle are determined by two factors: one is the velocity and the other is the location. The inertia weight PSO is the further development of the canonical PSO by adding an inertia weight in front of the velocity, and this technique is first proposed by Shi and Eberhart [32] with the details of the iteration paradigm shown below in Eq. 3:

$$\begin{cases} V_{i,G+1} \leftarrow \omega \cdot V_{i,G} + c_1 \cdot r_1 \cdot \left(X_{fb_i,G} - X_{i,G}\right) + c_2 \cdot r_2 \cdot \left(X_{gb,G} - X_{i,G}\right) \\ X_{i,G+1} \leftarrow X_{i,G} + V_{i,G} \end{cases} \tag{3}$$

where $V_{i,G}$ denotes the velocity of the i^{th} particle in the G^{th} generation, ω denotes the inertia weight of the velocity, $X_{fb_i,G}$ denotes the former best location of the i^{th} particle and $X_{gb,G}$ denotes the global best location of the population $\mathbf{P}$, c_1 and c_2 are two fixed constant values, $c_1 = c_2 = 2$, and r_1 and r_2 are two random values in the range $(0, 1)$.

Generally, iwPSO with a larger ω tends to have better exploration capacity while iwPSO with a smaller ω tends to have better exploitation capacity. In order to better explore the solution space at the earlier stage of the iteration while to better exploit a certain local area at the later part of the iteration, the iwPSO employs a linear decreasing function of iteration in the adaptation of ω rather than a fixed constant value. This can make a better balance between the exploration and exploitation of PSO. However, it can be seen that there are only four elements used during the evolution: the first three are the current position $X_{i,G}$, the current velocity $V_{i,G}$ and the former best location $X_{fb_i,G}$ of the i^{th} particle and the last is the global best location of the population $X_{gb,G}$. Obviously much information in the evolution was omitted and the velocity was also not restricted, all of which resulted in a poor performance of the canonical PSO algorithm.

2.2. The CLPSO algorithm

The PSO variants with global best topology tended to converge quickly into some local optima or the global best optimum, and the former proposed topologies, such as U-ring, Von Neumann, Star and Pyramid etc., had been tested with varying results. In order to improve the performance on multimodal objectives, the CLPSO, a new variant of PSO with comprehensive learning strategy, was proposed [12]. In CLPSO, each dimension of a particle only learns from the corresponding dimension of the exemplar which may be its own historical best or any particle's historical best. Which one should be selected as the exemplar, $X_{pbest_{f_i,G}}$, is determined by a probability Pc of the i^{th} particle. If a random number is larger than P_{c_i}, the corresponding dimension will learn from the i^{th} particle's historical best, otherwise, it will learn from the corresponding dimension of another particle's historical best. The update equation of CLPSO is presented in Eq. 4.

$$\begin{cases} V_{i,G}^d = \omega_i \cdot V_{i,G}^d + c \cdot r_{i,G} \cdot \left(X_{pbest_{f_i,G}}^d - X_{i,G}^d\right) \\ X_{i,G} = X_{i,G} + V_{i,G} \end{cases} \tag{4}$$

where $V_{i,G}^d$ denotes the d^{th} dimension of $V_{i,G}$, $X_{i,G}^d$ denotes the d^{th} dimension of $X_{i,G}$, c is a constant parameter and $r_{i,G}$ denotes a random value in $(0, 1)$. By the way, the probability P_c ranges from 0.05 to 0.5 and obeys Eq. 5, and ω_i obeys a linear reduction according to Eq. 6.

$$Pc_i = 0.05 + 0.45 \cdot \frac{e^{10 \cdot \frac{i-1}{ps-1}} - 1}{e^{10} - 1} \tag{5}$$

$$\omega = \omega_{\max} - \frac{gen}{\mathrm{gen}_{\max}} (\omega_{\max} - \omega_{\min}) \tag{6}$$

Here, *gen* denotes the current iteration number, gen_{max} is maximum number of iterations, $\omega_{\max}$ denotes the initial value of ω and $\omega_{\min}$ denotes the terminal value of ω. The default values of $\omega_{\max}$ and $\omega_{\min}$ in CLPSO are ω_{max}=0.9 and ω_{min}=0.4.

2.3. The DNLPSO algorithm

The DNLPSO algorithm was actually a further extension of the CLPSO algorithm for complicated and multi-modal optimization problems [13], therefore, there were some identical aspects and different aspects between them. Both CLPSO and DNLPSO employed other particles' historical best for the update of velocity, however, the other particles were selected by pair-wise competition in CLPSO while they were within a ring-topology in DNLPSO. The ring-topology in DNLPSO was also changed during the iteration by some intervals, and regrouping operation of the particles were conducted every 5 iterations for complicated problems and every 10 iteration for simple problems. The learning strategy from other particles' historical best was only activated when a random generated number was smaller than *Pc*, otherwise, the particle also leant from its personal historical best. The selection probability *Pc* in DNLPSO shared the same range [0,0.5] as the one in CLPSO but it obeyed a linear reduction with the equation given below:

$$Pc_i = 0.5 - 0.5 \cdot \frac{i-1}{gen_{\max} - 1} \tag{7}$$

where i denotes the iteration number. Moreover, the global best information was also used in the iteration equation of DNLPSO, which was given in Eq. 8:

$$\begin{cases} V_{i,G}^d = \omega \cdot V_{i,G}^d + c_1 \cdot r_1 \cdot \left(X_{pbest_{f_{i,G}}}^d - X_{i,G}^d\right) + c_2 \cdot r_2 \cdot \left(X_{gb,G}^d - X_{i,G}^d\right) \\ X_{i,G}^d = X_{i,G}^d + V_{i,G}^d \end{cases} \tag{8}$$

where the same symbols have the same meaning as mentioned before in the paper. The algorithm was verified on the selected benchmarks, however, we found that its performance was not satisfactory under the Congress on Evolutionary Computation (CEC) benchmarks [9].

2.4. The SLPSO algorithm

Generally, there are many different kinds of social behaviors in the real world, and the PSO variants mainly employ the global best and personal historical best information of the population. Rather than employing these information, the SLPSO algorithm [40] made a big modification in comparison with the classical PSO variants, and a novel social learning mechanism that each particle learned from any better particle of the population was proposed with the details shown in Eq. 9:

$$X_{i,G+1} = \begin{cases} X_{i,G} + V_{i,G+1}, & if\ p_{i,G} \leqslant P_i^L \\ X_{i,G} & otherwise \end{cases} \tag{9}$$

where $V_{i,G+1}$ satisfies:

$$\begin{cases} V_{i,G+1} = r_1 \cdot V_{i,G} + r_2 \cdot I_{i,G} + r_3 \cdot \epsilon \cdot C_{i,G} \\ I_{i,G} = X_{k,G} - X_{i,G} \\ C_{i,G} = \overline{X}_{center,G} - X_{i,G} \end{cases} \tag{10}$$

$X_{k,G}$ denotes a randomly selected better particle of the i^{th} particle in the G^{th} iteration, $\overline{X}_{center,G}$ denotes the center of the population, ϵ denotes the social influence, $\epsilon = 0.01 \cdot \frac{D}{100}$, and D is the dimension number. r_1, r_2 and r_3 are three random values in $(0, 1)$. P_i^L denotes the learning probability of the i^{th} particle, and it obeys Eq. 11:

$$P_i^L = \left(1 - \frac{i-1}{ps}\right)^{\ln\left(\lceil \frac{D}{100} \rceil\right)} \tag{11}$$

2.5. The EPSO algorithm

Generally, there was no single update equation of a PSO variant to tackle every optimization problem effectively and efficiently, therefore, Lynn and Suganthan proposed an Ensemble Particle Swarm Optimizer (EPSO) to meet various optimization demands [41]. In the EPSO algorithm, there are 5 basic update equations from iwPSO [32], CLPSO [12], fitness-distance-ratio based PSO (FDR-PSO) [43], self-organizing hierarchical PSO (HPSO-TVAC) [44] and distance-based locally informed PSO (LIPS) [45] respectively.

(A) **FDR-PSO**: The FDR-PSO algorithm was proposed to tackle the premature convergence of the iwPSO algorithm by incorporating the neighboring particles with best particle of fitness-distance-ratio. To be more exactly, the particle which

minimizes fitness-distance-ratio is selected as the neighbor particle, $X_{nb,G}$, in minimization optimization. The fitness-distance-ratio r is calculated according to Eq. 12:

$$r = \frac{f(X_{j,G}) - f(X_{i,G})}{|X^d_{j,G} - X^d_{i,G}|} \tag{12}$$

After all the neighbor particles are determined, we can use the following Eq. 13 for the updating of particles in FDR-PSO. The detailed equations are given as follows:

$$\begin{cases} V^d_{i,G} = \omega \cdot V^d_{i,G} + c_1 \cdot r_1 \cdot \left(X^d_{fb_i,G} - X^d_{i,G}\right) + c_2 \cdot r_2 \cdot \left(X^d_{gb,G} - X^d_{i,G}\right) \\ \qquad\qquad + c_3 \cdot r_3 \cdot \left(X^d_{nb,G} - X^d_{i,G}\right) \\ X_{i,G} = X_{i,G} + V_{i,G} \end{cases} \tag{13}$$

where $c_1 = 1, c_2 = 1, c_3 = 2$, and r_1, r_2, and r_3 are random values in $(0, 1)$.

(B) **HPSO-TVAC**: The HPSO-TVAC algorithm was proposed to improve the overall performance of PSO not only on unimodal objective but also on multimodal objectives through two aspects of enhancements. The first enhancement is that it proposed a Time-Varying Acceleration Coefficient (TVAC) strategy for the effective control of local search and efficient convergence of global search; The second is a concept of self-organizing hierarchical particle swarm optimizer (HPSO). The detailed equation of these two components are presented below in Eq. 14 and Eq. 15 respectively.

$$\begin{cases} c_1 = 2.5 - 2 \cdot \frac{G}{G_{\max}} \\ c_2 = 0.5 + 2 \cdot \frac{G}{G_{\max}} \end{cases} \tag{14}$$

$$\begin{cases} V_{i,G} = c_1 \cdot r_1 \cdot (X_{fb_i,G} - X_{i,G}) + c_2 \cdot r_2 \cdot (X_{gb,G} - X_{i,G}) \\ X_{i,G} = X_{i,G} + V_{i,G} \end{cases} \tag{15}$$

(C) **LIPS**: The Locally Informed Particle Swarm (LIPS) algorithm aimed at tackling multi-modal optimization problem on the observation that distance-based topology could form different stable niches and consequently find the peaks of these different niches. Therefore, it was more capable of tackling multi-modal optimization problems rather than the Fully-Informed Particle Swarm (FIPS) algorithm [19]. The details of the update equation is given below:

$$\begin{cases} V_{i,G} = \chi * (V_{i,G} + \varphi(P_{i,G} - X_{i,G})) \\ X_{i,G} = X_{i,G} + V_{i,G} \end{cases} \tag{16}$$

where $P_i = \frac{\sum_{j=1}^{N}\left(\varphi_j * X_{nb_j,G}\right)/N}{\varphi}$, χ is the constriction coefficient ($\chi = 0.7298$), φ denotes the acceleration weight and it equals to the summation of φ_j, φ_j is a random number generated according to a uniform distribution in [0,4.1/N], and N is the size of the neighborhood.

The EPSO algorithm employed a self-adaptive selection strategy to determined which update equation was used for the particles in the larger sub-population each generation. In the strategy, a fixed number of learning period was used and the selection probability of each update equation after the learning period was calculated according to Eq. 17:

$$\begin{cases} p_{k,G} = \frac{S_{k,G}}{\sum_{k=1}^{K} S_{k,G}} \\ S_{k,G} = \frac{\sum_{gen=G-LP}^{G-1} ns_{k,G}}{\sum_{gen=G-LP}^{G-1} ns_{k,G} + \sum_{gen=G-LP}^{G-1} nf_{k,G}} + \epsilon \end{cases} \tag{17}$$

where $S_{k,G}$ represents the success rate of the k^{th} PSO strategy in the G^{th} generation of the larger sub-population, and K is the total number of strategies in the sub-population, $ns_{k,G}$ denotes the number of success update while $nf_{k,G}$ denotes the number of failure update of the k^{th} PSO strategy in the G^{th} generation, LP denotes the number of learning generations, and ϵ=0.01 is used to avoid the possible null success rate. The self-adaptive selection strategy was not new and it had been employed in SaDE [46] algorithm in 2009.

2.6. The MPSO algorithm

Similar to the HPSO-TVAC algorithm, the MPSO algorithm also improved the PSO algorithm through two aspects: one was the enhancement of the inertia weight and the other is the enhancement of update equations [42]. In MPSO, a chaos-based non-linear inertia weight was proposed to balance the exploration and exploitation. Meanwhile, an adaptive strategy based update equation of particle was also proposed to strengthen the capacity of PSO in tackling complex problems. The details of the inertial weight and the update equation of particle are given in Eq. 18 and Eq. 20 respectively.

$$\omega(gen) = r(gen) \cdot \omega_{\min} + \frac{(\omega_{\max} - \omega_{\min}) \cdot gen}{gen_{\max}} \tag{18}$$

where $r(gen)$ can be generated according to Eq. 19:

$$r(gen+1) = 4 \cdot r(gen) \cdot (1 - r(gen)) \tag{19}$$

The initial number $r(0)$ is a random number and it obeys $r_0 \notin \{0, 0.25, 0.5, 0.75, 1\}$.

$$\begin{cases} V_{i,G} = \omega \cdot V_{i,G} + r_1 \cdot c_1 \cdot \left(X_{sb_i,G} - X_{i,G}\right) + r_2 \cdot c_2 \cdot \left(\overline{X}_{center,G} - X_{i,G}\right) \\ X_{i,G} = \begin{cases} \omega \cdot X_{i,G} + (1-\omega) \cdot V_{i,G+1} + X_{gb,G}, \ p_i > rand \\ X_{i,G} + V_{i,G+1}. \ otherwise \end{cases} \end{cases} \tag{20}$$

Here $X_{sb_i,G}$ denotes the stochastic learning particle of the i^{th} individual, and $\overline{X}_{center,G}$ is the same as the one in SLPSO and it denotes the center of the population.

3. The novel PSO-sono algorithm

In this section, we present the novel PSO variant in details, and we name it **PSO-sono** because the algorithm aims at providing excellent performance on **s**ingle-**o**bjective **n**umerical **o**ptimization. The whole description of **PSO-sono** can be divided into three parts: the first part discusses the hybrid paradigms of our novel PSO algorithm; the second part presents the adaptation schemes both for the ratio of each paradigm and the coefficients which constrict the cognitive component and the social component; and the third part presents a fully-informed search scheme around the global optimum during each generation, which can help the algorithm to jump out a certain local optimum during the whole iteration.

3.1. The update paradigm in our PSO algorithm

The update paradigms in the iwPSO algorithm and the ccPSO algorithm have excellent performance on many uni-modal objectives, therefore, the incorporation of these paradigms can improve the performance on many uni-modal objectives. Because adaptations of coefficients are also involved in our algorithm, it is better to choice the paradigm in the iwPSO algorithm rather than the ccPSO algorithm because parameters in iwPSO are of less dependency. One of the update paradigm in our algorithm is given in Eq. 21 below:

$$\begin{cases} V_{i,G+1} \leftarrow \omega \cdot V_{i,G} + c_{1,G} \cdot r_1 \cdot \left(X_{fb_i,G} - X_{i,G}\right) + c_{2,G} \cdot r_2 \cdot \left(X_{gb,G} - X_{i,G}\right) \\ X_{i,G+1} \leftarrow X_{i,G} + V_{i,G} \end{cases} \tag{21}$$

where $\omega, c_{1,G}$ and $c_{2,G}$ obey some adaptation schemes that will be discussed in the next subsection.

As is mentioned above, the iwPSO and ccPSO often had premature convergence because of lack of diversity, and a good way to tackle this weakness is to incorporate more information of the sociology. Based on this observation, the social behavior proposed in the SLPSO algorithm is also incorporated into our algorithm. The detailed equation of the other update paradigm in our algorithm is given in Eq. 22:

$$\begin{cases} V^d_{i,G+1} = r_0 \cdot V^d_{i,G} + r_1 \cdot I^d_{i,G} + r_2 \cdot \epsilon \cdot C^d_{i,G} \\ X_{i,G+1} = X_{i,G} + V_{i,G+1} \end{cases} \tag{22}$$

where $I_{i,G+1}$ and $C_{i,G+1}$ satisfy:

$$\begin{cases} I_{i,G} = X_{k,G} - X_{i,G} \\ C_{i,G} = \overline{X}_{center,G} - X_{i,G} \end{cases} \tag{23}$$

By the way, the values of r_0, r_1 and r_2 in Eq. 21 and Eq. 22 of the hybrid paradigms are generated in the same way according to the uniform distribution of (0,1), but each parameter/variable in the D-dimensional velocity employs a new generated r_0 while all the parameters/variables in the D-dimensional position employ the same values of r_1 and r_2.

3.2. The adaptation schemes in our algorithm

(A) **The adaptation scheme of ratio**: In our algorithm, the whole particles are divided into two groups, the better-particle-group and the worse-particle-group by sorting the particles regarding to their fitness values. Particles in the better-particle-group employ Eq. 21 while particles in the worse-particle-group employ Eq. 22 in iteration. The number of particles in the two groups are dynamically changed during the iteration, and a ratio r is used for denoting the percentage of the better particles in the whole population, and it can be calculated according to Eq. 24 when the number of success particles is more than zero in each iteration:

$$r = \frac{ns_b}{ns_b + ns_w} \tag{24}$$

where ns_b and ns_w denote the numbers of success particles in the better-particle-group and worse-particle-group respectively. Moreover, a truncation readjustment of r is involved when its value is larger than 0.9 or smaller than 0.1.

(B) **The adaptation scheme of constriction coefficients**: The balance between exploration and exploitation play a key role in the optimization performance of PSO variants. As it is known to all that the parameter c_1 constricts the cognitive part while c_2 constricts the social part of the paradigm. Particles in a PSO variant with a large c_1 and a small c_2 at the beginning of iteration tend to move around the population instead of moving toward the global optimum, therefore, the premature convergence can be tackled. Based on this observation, the adaptation scheme of these two parameters in our PSO-sono are given below:

$$\begin{cases} \omega(G) = \omega_{\max} - \frac{G}{G_{\max}}(\omega_{\max} - \omega_{\min}) \\ c_{1,G} = 2.5 - 2 \cdot \frac{G}{G_{\max}} \\ c_{2,G} = 0.5 + 2 \cdot \frac{G}{G_{\max}} \end{cases} \tag{25}$$

where G denotes the current number of iterations, $\omega_{\max}$ and $\omega_{\min}$ denote the initial and terminal inertia weight respectively, and $G_{\max}$ denotes the maximum iteration number.

3.3. Fully-informed search scheme

Generally, many of the PSO variants usually have premature convergence weakness especially when tackling some multimodal objectives. In this part, we present the fully-informed search scheme of our PSO-sono algorithm. The thought of the fully-informed search scheme is that the knowledge of the whole population may draw the global best particle out of the current location and consequently avoiding premature convergence. The novel fully-informed search scheme is given in Eq. 26, and this scheme is one of the key components of our algorithm in each iteration.

$$X_{gbest,G} = \begin{cases} X_{gbest,G} & f(X_{gbest,G}) < f(X_{FIS,G}) \\ X_{FIS,G} & otherwise \end{cases} \tag{26}$$

where $X_{FIS,G}$ is calculated according to Eq. 27:

$$X_{FIS,G} = \begin{cases} X_{gbest,G} \cdot cos\left(1 - \frac{G}{G_{\max}}\right) + \sigma, \ if\ rnd < 0.5 \\ X_{gbest,G} \cdot sin\left(1 - \frac{G}{G_{\max}}\right) + \sigma, \ otherwise \end{cases} \tag{27}$$

The σ which can be considered as the fully-informed vector can be calculated via Eq. 28:

$$\begin{cases} \overline{X}_{center,G} = \frac{1}{PS}\sum_{i=1}^{PS} X_{i,G} \\ var_i = (X_{i,G} - \overline{X}_{center,G}) + \varphi \cdot (P_{i,G} - X_{i,G}) \\ \sigma = \frac{1}{ps}\sum_{i=1}^{PS} var_i \end{cases} \tag{28}$$

where $P_i = \frac{\sum_{j=1}^{N}\left(\varphi_j * X_{nb_j,G}\right)/N}{\varphi}$ ($X_{nb_j,G}$ is the neighbor particle of $X_{i,G}$ in the ring topology) and φ denotes the acceleration weight and it equals to the summation of φ_j, $\varphi = \sum_{j=1}^{N}\varphi_j$, φ_j is a random number generated according to a uniform distribution in [0,4.1/N], and N is the size of the neighborhood. The pseudo code of our novel PSO-sono algorithm is also presented here in Algo. 1.

Algorithm 1: Pseudo code of our novel PSO-sono algorithm.

Input: Solution space $[X^D_{\min}, X^D_{\max}]$, Velocity bound $[V^D_{\min}, V^D_{\max}]$, maximum number of function evaluation $nfe_{\max}$, objective function $f(X)$.
Output: Number of function evaluation nfe, optimum location $X_{gbest,G}$, best fitness value $f(X_{gbest,G})$.

```
r = 0.5, G = 1.
for i = 1 : PS do
  Initialize the i^th particle, X_{i,G}.
  Calculate the fitness value f(X_{i,G}).
end for
nfe = ps;
Label X_{gbest,G}, output nfe, f(X_{gbest,G}).
while nfe < nfe_max do
  Sort the population in descending order.
  Calculate the parameters according to Eq. 25.
  Separate it into two groups according to the ratio.
  Update particles in 1^st-group according to Eq. 21.
  Update particles in 2^nd-group according to Eq. 22.
  for i = 1 : PS do
    Initialize the i^th individuals, X_{i,G}.
    Calculate the fitness value f(X_{i,G}).
  end for
  nfe = nfe + ps.
  Update the parameters according Eq. 24.
  Fully-informed search according to Eq. 26.
  nfe = nfe + 1.
  Update X_{gbest,G} and f(X_{gbest,G});
end while
return nfe, X_{gbest,G} and f(X_{gbest,G}).
```

4. Experiment analysis

In this part, we present the detailed analysis of the PSO-sono algorithm, and all the experiments were conducted on Matlab 2011b version of a personal computer with an Intel(R) Core(TM) i5-8265U 1.8 GHz CPU and Microsoft Windows 10 Enterprise 64-bit Operating System. Based on the observation that employing a smaller test suite in algorithm validation may have over-fitting problem, here we employed 88 benchmarks from CEC2013, CEC2014 and CEC2017 test suites for real-parameter single-objective optimization in the verification of the algorithm. Among the 88 benchmarks, there are 28 benchmarks from CEC2013 test suite, 30 benchmarks from CEC2014 test suite and 30 benchmarks from CEC2017 test suites. Benchmarks from CEC2013 test suite are denoted as f_{a_1}–$f_{a_{28}}$, benchmarks from CEC2014 test suite are denoted as f_{b_1}–$f_{b_{30}}$, and benchmarks from CEC2017 test suite are denoted as f_{c_1}–$f_{c_{30}}$ in our test suite. All these benchmarks can be divided into four groups: the unimodal function group contains f_{a_1}–f_{a_5}, f_{b_1}–f_{b_3} and f_{c_1}–f_{c_3}; the multi-modal function group contains f_{a_6}–$f_{a_{20}}$, f_{b_4}–$f_{b_{16}}$ and f_{c_4}–$f_{c_{10}}$; the hybrid function group contains $f_{b_{17}}$–$f_{b_{22}}$, $f_{c_{11}}$–$f_{c_{20}}$; and the composition group contains $f_{a_{21}}$–$f_{a_{28}}$, $f_{b_{23}}$–$f_{b_{30}}$, and $f_{c_{21}}$–$f_{c_{30}}$. The maximum number of function evaluation in algorithm validation is $10000 \cdot D$, and the statistics, the mean and standard deviation of the fitness error $f - f^*$, are calculated from the results of 51 runs.

4.1. Optimization accuracy

In the first-group comparison, there are 6 PSO algorithms taken into comparison, and they are the inertia weight PSO algorithm (iwPSO) [32], the Comprehensive Learning PSO algorithm (CLPSO) [12], the Dynamic Neighborhood Learning PSO algorithm (DNLPSO) [13], the Social Learning PSO algorithm (SLPSO) [40], the Ensemble PSO algorithm (EPSO) [41] and the MPSO algorithm [42], all of which are closely related algorithms with our PSO-sono algorithm. The parameter settings of these algorithms are the defaults ones recommended by the authors, and they are listed in Table 1. The comparison results under f_{a_1}–$f_{a_{28}}$ (CEC2013), f_{b_1}–$f_{b_{30}}$ (CEC2014), f_{c_1}–$f_{c_{30}}$ (CEC2017) of our test suite on 10D optimization, 30D optimization and 50D optimization were listed in Table S-1 – Table S-9 of the supplementary file respectively. There are two parts in these results: the first part is the "Mean/Std" (mean and standard deviation) of 51 runs and the second part is the overall

Table 1
The default parameter settings of these PSO variants.

Algo.	The default parameter settings
iwPSO [11]	$PS = 100$, $c_1 = c_2 = 2.0$, $iw \in [0.4, 0.9]$
CLPSO [12]	$PS = 50$, $c_1 = c_2 = 1.49455$, $iw \in [0.4, 0.9]$, $Pc \in [0, 0.5]$, $stay\_num = 7$
DNLPSO [13]	$PS = 100$, $c_1 = c_2 = 1.49445$, $iw \in [0.4, 0.9]$, $Pc \in [0.05, 0.45]$, $m = 3$, $g = 5$
SLPSO [40]	$M = 100, PS = M + \lfloor \frac{D}{10} \rfloor$, $\epsilon = \frac{D}{M} \cdot 0.01$, $PL \in [0, 1]$
EPSO [41]	$PS = 100$, default settings of its components
MPSO [42]	$PS = 100$, $iw \in [0.4, 0.9]$, $cw = 4 \cdot r \cdot (1 - r)$, r is a random value and $r \in (0, 1)$
Our algorithm	$PS = 100$, $iw \in [0.4, 0.9]$, $\epsilon = \frac{D}{PS} \cdot 0.01$, $r = 0.5$, Uring-topology

performance of all the 51 samples evaluated under Wilcoxon's Signed Rank test with the significant level $\alpha = 0.05$, and the evaluations are denoted by symbols "$>$", "$\approx$" and "$<$" which mean "Better performance", "Similar performance" and "Worse performance" respectively. Symbols "$>/\approx/<$" at the bottom of each table represent the sums of the comparison results on each benchmark under sign test, and fitness error smaller than $eps = 2.2204\text{e}{-16}$ is considered as 0 in these tables.

From the comparison results we can see that: (1) our PSO-sono algorithm performs no worse than all the competitors including iwPSO [11], CLPSO [12], DNLPSO [13], SLPSO [40], EPSO [41] and MPSO [42] on 10D, 30D and 50D optimization. (2) our PSO-sono algorithm obtains big performance improvement in comparison with iwPSO and DNLPSO on all the test suites including CEC2013, CEC2014 and CEC2017; it also obtains big performance improvement in comparison with CLPSO and MPSO on CEC2013 and CEC2014 test suites; and it also obtains big performance improvement in comparison with SLPSO and EPSO on CEC2013 test suite. (3) our PSO-sono algorithm is more likely to obtain big performance improvement under the CEC2013 test suite in comparison with the CEC2014 and CEC2017 test suites. (4) our PSO-sono algorithm obtains more performance improvements on 10D optimization in comparison with 30D and 50D optimization. 5) our PSO-sono algorithm performs excellent on 30D and/or 50D optimization under a certain test suite in comparison with a competitor while performing no worse under the other test suites.

As one of the reviewers suggested, we also examined the performance of our PSO-sono on higher dimensional optimization under the benchmarks f_{c_1}–$f_{c_{30}}$ of our test suite, and the comparison results are given in Table S-10 of the supplementary file. It can be seen from these results that our PSO-sono algorithm obtains 22 performance improvements in comparison with iwPSO, obtains similar performance in comparison with CLPSO and MPSO, obtains 30 performance improvements in comparison with DNLPSO, obtains 18 performance improvements in comparison with SLPSO, and obtains 17 performance improvements in comparison with EPSO. Again, our PSO-sono algorithm is still competitive with the contrasted PSO variants from the perspective of 100D optimization.

Table 2 presents a summary of the comparison results (according to the results in Table S-1 to Table S-10 of the supplementary file) of a certain algorithm versus our PSO-sono algorithm under our test suite containing 88 benchmarks. We can see from the summary that the PSO-sono algorithm obtains 213 performance improvements out of 294 cases in comparison with iwPSO, obtains 165 performance improvements out of 294 cases in comparison with CLPSO, 290 out of 294 cases in comparison with DNLPSO, 163 out of 294 cases in comparison with SLPSO, 156 out of 294 cases in comparison with EPSO and 173 out of 294 cases in comparison with MPSO.

As one of the reviewers suggested, we also examined the performance of our PSO-sono in comparison with two extra PSO variants including the PPSO algorithm [47] and the TSLPSO algorithm [48] under benchmarks f_{a_1}–$f_{a_{28}}$ of our test suite. The three PSO variants including PPSO, TSLPSO and our PSO-sono are compared on 10D, 30D and 50D optimization, and the comparison results are given in Table S-11 of the supplementary file with a summary shown below in Table 3. From these results we can see that our PSO-sono obtains 14 performance improvements, 17 performance improvements and 14 performance improvements on 10D, 30D and 50D respectively in comparison with TSLPSO algorithm and it also obtains 23 performance

Table 2
Summary of the comparison results from Table 2 to Table 9 of the supplementary file between our PSO-sono algorithm and the other contrasted algorithms under the CEC2013, CEC2014 and CEC2017 benchmarks.

A given algorithm versus our PSO-sono algorithm											
Test suit:	CEC2013			CEC2014			CEC2017				All
$>/\approx/<$	D = 10	D = 30	D = 50	D = 10	D = 30	D = 50	D = 10	D = 30	D = 50	D = 100	$\sum$
iwPSO	8/1/19	6/0/22	7/0/21	8/0/22	11/0/19	10/0/20	7/0/23	8/0/22	7/0/23	8/0/22	80/1/213
CLPSO	11/1/16	9/1/18	10/0/18	12/0/18	12/0/18	15/0/15	15/0/15	14/0/16	14/0/16	15/0/15	127/2/165
DNLPSO	0/0/28	2/0/26	1/0/27	0/0/30	0/0/30	0/0/30	1/0/29	0/0/30	0/0/30	0/0/30	4/0/290
SLPSO	7/2/19	13/2/13	9/0/19	13/0/17	15/0/15	15/0/15	14/0/16	14/0/16	15/0/15	12/0/18	127/4/163
EPSO	10/1/17	12/0/16	14/0/14	14/1/15	14/0/16	15/0/15	15/0/15	14/0/16	15/0/15	13/0/17	136/2/156
MPSO	11/3/14	10/0/18	13/0/15	4/0/26	13/0/17	14/0/16	8/0/22	15/0/15	15/0/15	15/0/15	118/3/173

Table 3
Summary of the comparison results between our PSO-sono algorithm and the other contrasted algorithms under the CEC2013 benchmarks on 100D optimization.

Summary of the comparison results in Table S-11.			
Test suite:	CEC2013		
>/≈/<	D = 10	D = 30	D = 50
TSLPSO	13/1/14	11/0/17	14/0/14
PPSO	4/1/23	7/0/21	7/0/21

improvements, 21 performance improvements and 21 performance improvements on 10D, 30D and 50D respectively in comparison with PPSO algorithm. Obviously, our PSO-sono algorithm is still competitive with these two PSO variants. To summarize, our novel algorithm secures an overall better performance under our test suite containing 88 benchmarks from optimization accuracy perspective.

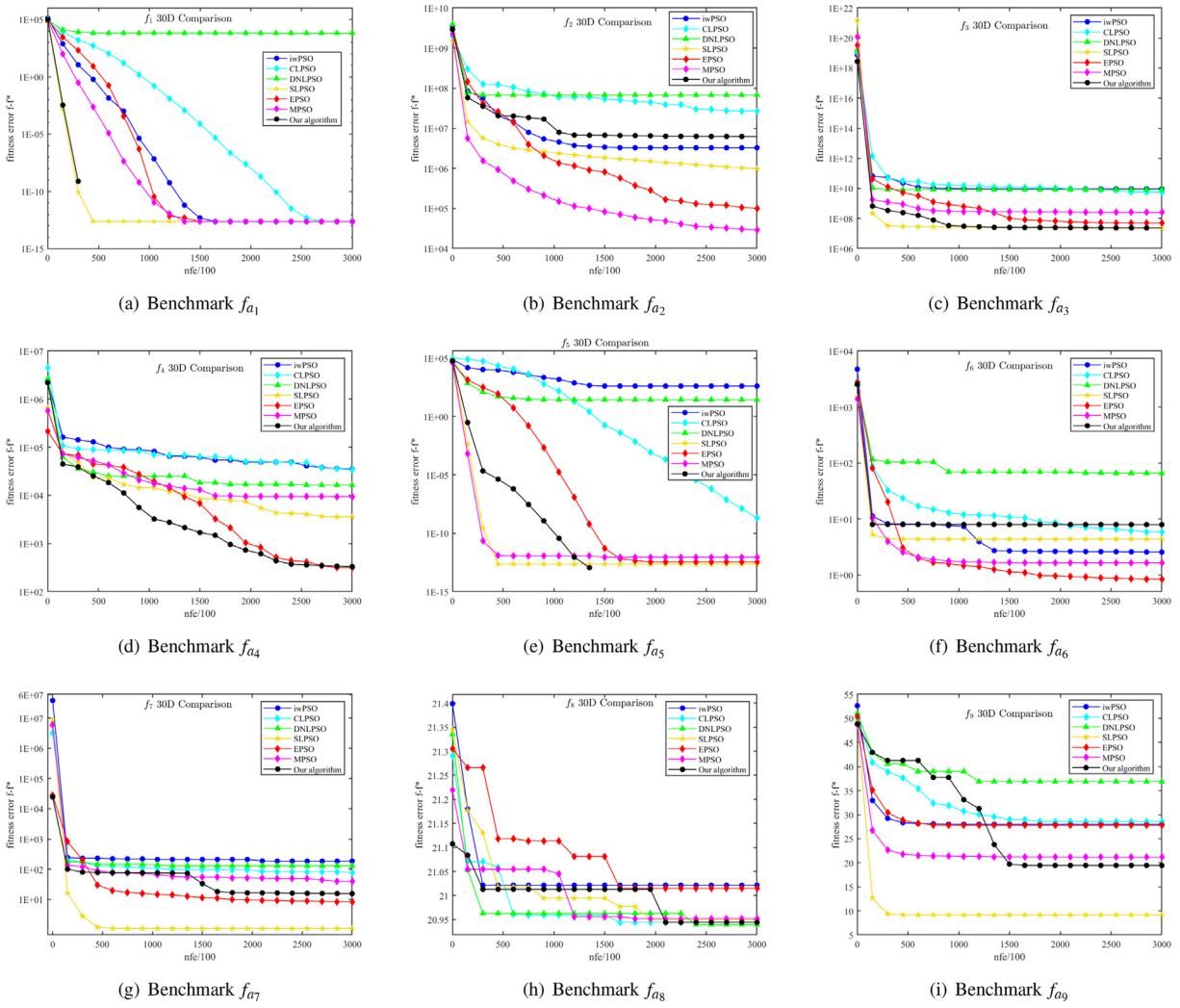
4.2. Convergence analysis

The novel PSO-sono algorithm is also examined from convergence speed perspective, and the median value of the 51-run of a certain algorithm is selected out and the corresponding trajectory of the global best during the iteration is considered as the convergence curve which is depicted in Fig. 1–3 under benchmarks $f_{a_1}-f_{a_{28}}$ of our test suite on 30D optimization. It can be seen from the convergence curves that our algorithm obtains better or similar performance in comparison with iwPSO on $f_{a_1},f_{a_3}-f_{a_5},f_{a_7}-f_{a_{13}},f_{a_{15}},f_{a_{17}}-f_{a_{21}},f_{a_{24}}-f_{a_{28}}$; obtains better or similar performance in comparison with CLPSO on $f_{a_1}-f_{a_5},f_{a_7}-f_{a_{10}},f_{a_{12}},f_{a_{13}},f_{a_{15}},f_{a_{16}},f_{a_{18}}-f_{a_{21}},f_{a_{24}}-f_{a_{26}}$ and $f_{a_{28}}$; obtains better or similar performance in comparison with DNLPSO on $f_{a_1}-f_{a_7}$ and $f_{a_9}-f_{a_{28}}$; obtains better or similar performance in comparison with SLPSO on $f_{a_1},f_{a_4},f_{a_5},f_{a_8},f_{a_{10}},f_{a_{12}},f_{a_{13}},f_{a_{15}},f_{a_{17}}-f_{a_{21}}$ and $f_{a_{25}}-f_{a_{28}}$; obtains better or similar performance in comparison with EPSO on $f_{a_1},f_{a_3},f_{a_5},f_{a_8},f_{a_9},f_{a_{11}}-f_{a_{13}},f_{a_{15}},f_{a_{17}}-f_{a_{21}}$ and $f_{a_{24}}-f_{a_{28}}$; obtains better or similar performance in comparison with MPSO on $f_{a_1},f_{a_3}-f_{a_5},f_{a_7}-f_{a_{10}},f_{a_{12}}-f_{a_{21}},f_{a_{25}},f_{a_{26}},f_{a_{28}}$. We also summarize these results in Table 4 presenting the objectives on which the proposed algorithm obtains best or tier best performance in comparison with a given algorithm. Furthermore, we also summarize the objectives on which a certain algorithm obtains the best or the tier performance on benchmarks f_{a_1}-$f_{a_{28}}$ of our test suite in Table 5, and we can see that the proposed algorithm is still competitive with the other contrasted well-known PSO variants in terms of best or tier best of convergence speed.

From Fig. 1–3 we also can see that our PSO-sono algorithm is not the worst algorithm on any benchmark of the test suite from the convergence perspective of view, and it performs relatively worse on the benchmarks $f_{a_2},f_{a_6},f_{a_{14}},f_{a_{22}}$ and $f_{a_{23}}$ because it only can beat CLPSO and DNLPSO on f_{a_2}, can beat DNLPSO on f_{a_6}, beat DNLPSO and MPSO on $f_{a_{14}}$ and beat DNLPSO on $f_{a_{22}}$ and $f_{a_{23}}$. Among these benchmarks, f_{a_2} is a rotated benchmark from the uni-modal function group which contains benchmarks f_{a_1}–f_{a_5} and f_{a_2} also has the same non-separable characteristic as benchmarks f_{a_3} and f_{a_4} in this group. f_{a_6} is a rotated benchmark from the multi-modal function group which contains benchmarks f_{a_6}–$f_{a_{20}}$ and it also has the same non-separable characteristic as f_{a_7}–$f_{a_{10}}$ and $f_{a_{12}}$–$f_{a_{20}}$. f_{14} also has the same asymmetrical characteristic as f_{a_7}–f_{a_9},$f_{a_{12}}$–$f_{a_{18}}$ and $f_{a_{20}}$ in the multi-modal function group. $f_{a_{22}}$ and $f_{a_{23}}$ are from the same composition function group, nevertheless, $f_{a_{22}}$ is a separable function while $f_{a_{23}}$ is a non-separable function. To summarize, we can not conclude that the PSO-sono algorithm performs good on separable functions rather than non-separable functions, or performs good on symmetrical function rather than asymmetrical functions, or performs good on uni-modal function or multi-modal function or composition functions. It can be concluded that our PSO-sono algorithm is very competitive with these compared algorithms, and the performance of our PSO-sono algorithm is highly dependent on the specifics of a certain objective.

4.3. The fully-informed search scheme

In order to further investigate the effectiveness of the fully-informed search scheme, experiments are conducted between the PSO-sono with fully-informed search scheme (the default) and the PSO-sono without fully-informed search scheme under the benchmarks f_{a_1}–$f_{a_{28}}$ of our test suite on 30D optimization. The results are given in Table 6, and it can be seen that PSO-sono with fully-informed search scheme obtains 18 performance improvements in comparison with PSO-sono without fully-informed search scheme. Moreover, big performance improvements can be obtained nearly on all the unimodal function group containing f_{a_1}–f_{a_5}, big performance improvements can be obtained on $f_{a_7},f_{a_8},f_{a_{10}},f_{a_{11}},f_{a_{13}},f_{a_{15}}$–$f_{a_{17}}$ and $f_{a_{19}}$ of the basic multi-modal function group, and big performance improvements can be obtained on $f_{a_{22}}$–$f_{a_{25}}$ and $f_{a_{27}}$ of the composition function group.



(a) Benchmark f_{a_1} (b) Benchmark f_{a_2} (c) Benchmark f_{a_3}

(d) Benchmark f_{a_4} (e) Benchmark f_{a_5} (f) Benchmark f_{a_6}

(g) Benchmark f_{a_7} (h) Benchmark f_{a_8} (i) Benchmark f_{a_9}

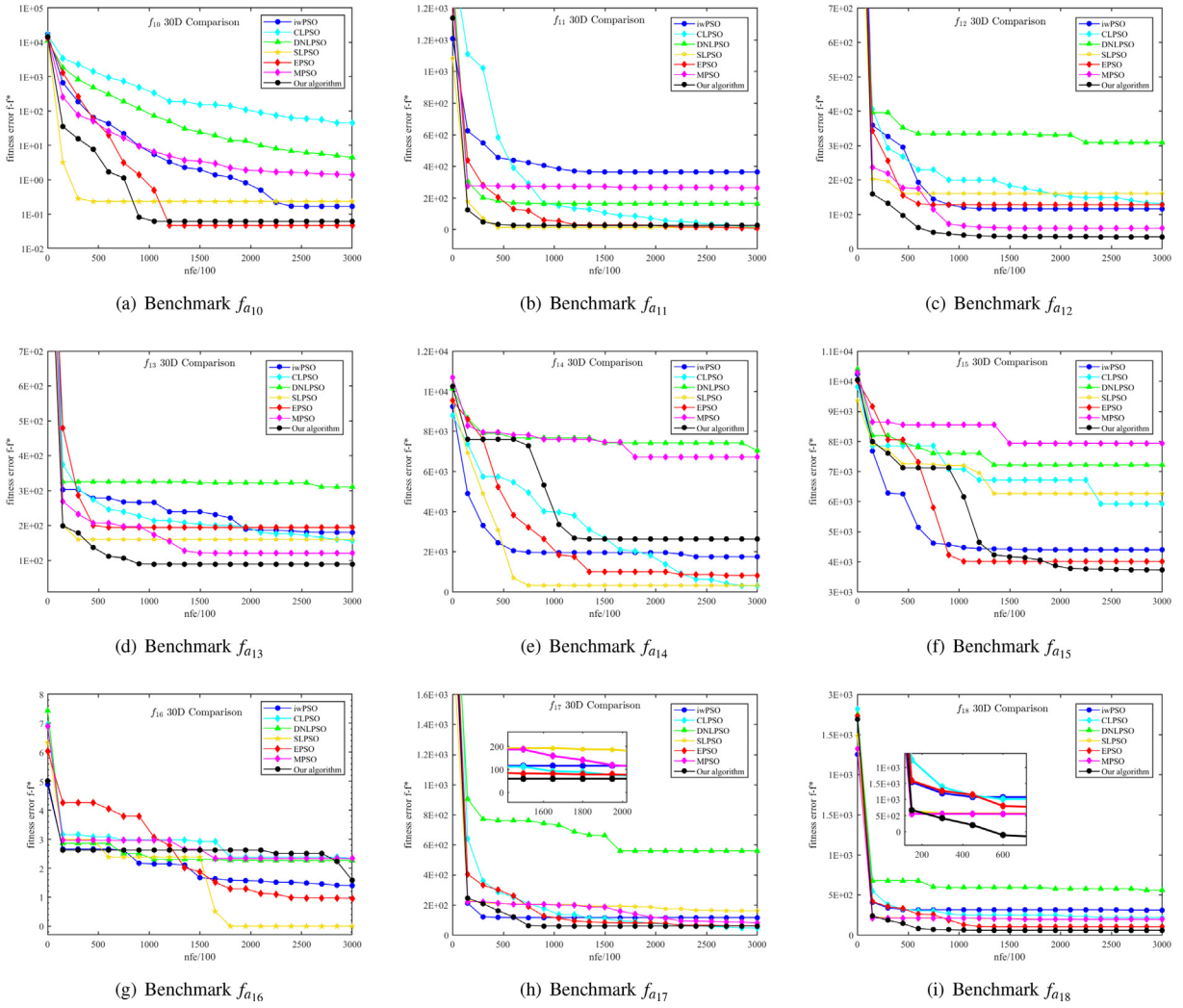
Fig. 1. Here presents the convergence speed comparison by employing the median value of 51 runs obtained by each algorithm on 30-D optimization under benchmarks f_{a_1}–$f_{a_{28}}$ of our test suite. There are total 28 comparison figures and the first 9 figures are presented here.

4.4. Time complexity analysis

We also examine the time complexity of our PSO-sono algorithm with these contrasted PSO variants in this part. The comparison of the time complexity was conducted according to the recommendation of CEC2013 competition, and results are presented in Table 7. T_0 denotes the time consumption of basic arithmetic expressions in the recommendation of CEC2013 competition, T_1 denotes the time consumption of 200000 function evaluations for 30D optimization on benchmark 14 of the CEC2013 test suite, and T_2 denotes the overall cost of a certain algorithm optimizing f_{14}. We run 21 times to get the average $\widehat{T}_1$ and $\widehat{T}_2$, and then $\frac{\widehat{T}_2-\widehat{T}_1}{T_0}$ are calculated as the time complexity of each algorithm. From the results we can see that our algorithm consumes more time than iwPSO, SLPSO, MPSO, TSLPSO and PPSO, moreover, our algorithm is much better than CLPSO, DNLPSO and EPSO from the time consumption perspective of view.

5. Conclusion

In this paper, we propose a new PSO variant, namely PSO-sono, aiming at improving the performance of PSO on single-objective numerical optimization. The novel algorithm has three contributions, the first one is that a sorted particle swarm



(a) Benchmark $f_{a_{10}}$ (b) Benchmark $f_{a_{11}}$ (c) Benchmark $f_{a_{12}}$

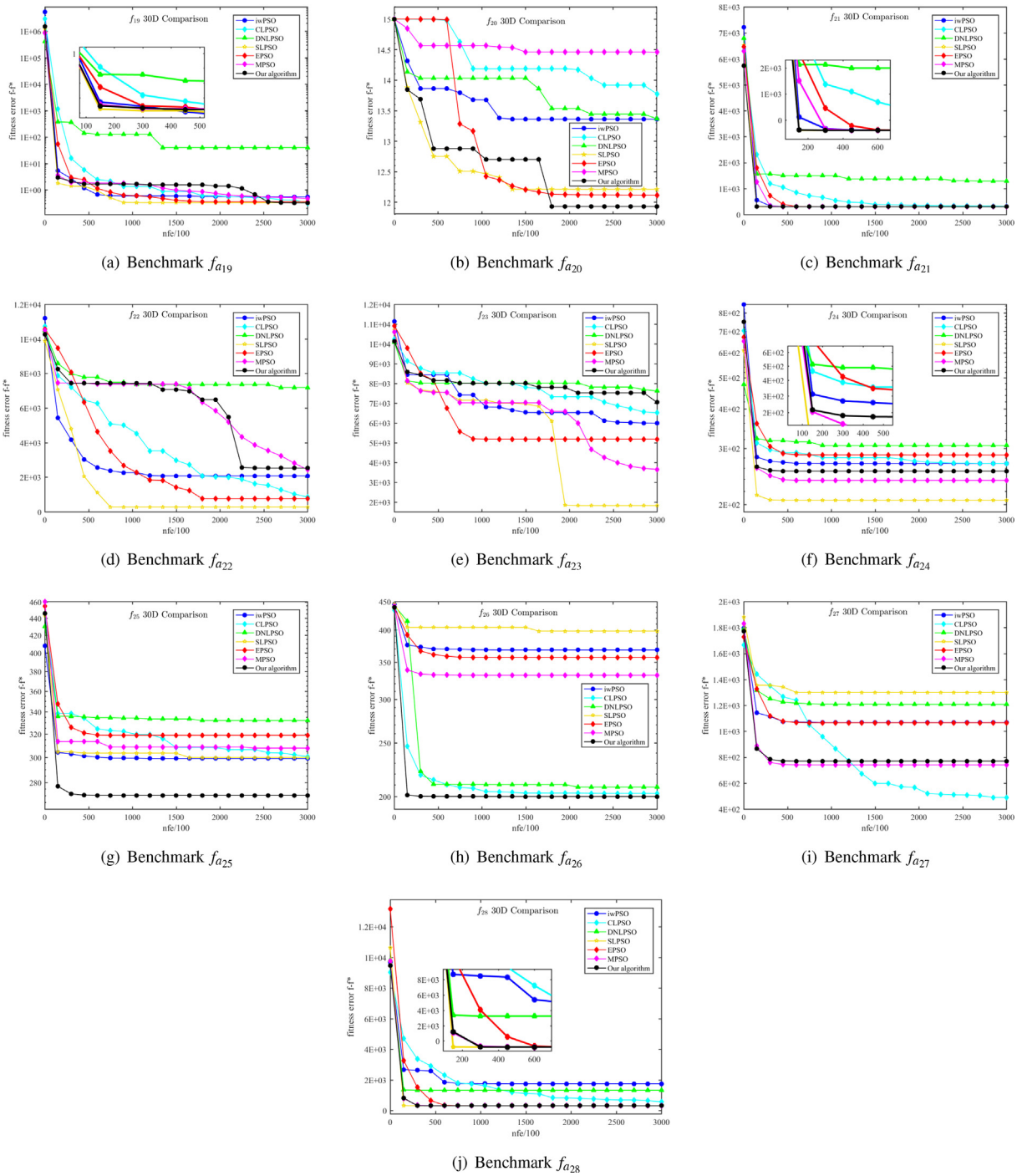
(d) Benchmark $f_{a_{13}}$ (e) Benchmark $f_{a_{14}}$ (f) Benchmark $f_{a_{15}}$

(g) Benchmark $f_{a_{16}}$ (h) Benchmark $f_{a_{17}}$ (i) Benchmark $f_{a_{18}}$

Fig. 2. As a continued part from Fig. 1, convergence comparisons on benchmarks $f_{a_{10}}$-$f_{a_{18}}$. are given here.

with hybrid paradigms is proposed. By employing an adaptive selection rate of each paradigm, the PSO-sono algorithm can achieve a good use of the advantages of all the paradigms. The second contribution is that novel adaptation schemes both for the coefficients and the selection rate were presented, and these adaptive schemes can help the algorithm to obtain a good balance between exploitation and exploration of different paradigms. The final contribution is that a fully-informed search scheme is proposed, and this technique can help the algorithm to exploit the knowledge of the whole population, and then help to jump out some local optima regarding the current population.

We also investigate the optimization accuracy, convergence speed, higher dimensional optimization capacity and time complexity of the novel PSO-sono algorithm as well as the brand new fully-informed search scheme of it on a large test suite containing 88 benchmarks from CEC2013, CEC2014 and CEC2017 test suites for single-objective numerical optimization, and the results approved the superiority of our algorithm in comparison with several famous PSO variants. According to the associated editor's suggestion, we also presented extra comparisons (in the supplementary file) between CLPSO with 100 population size and our algorithm, which also supported the superiority of our PSO-sono. In the near future, we'll test the efficiency of some of the components, e.g. the fully-informed search scheme, on other famous algorithms such as DE variants, CMAES variants, QUATRE variants [49,50] and make comparisons with them.



(a) Benchmark f_{a19} (b) Benchmark f_{a20} (c) Benchmark f_{a21}

(d) Benchmark f_{a22} (e) Benchmark f_{a23} (f) Benchmark f_{a24}

(g) Benchmark f_{a25} (h) Benchmark f_{a26} (i) Benchmark f_{a27}

(j) Benchmark f_{a28}

Fig. 3. As a continued part from Fig. 2, convergence comparisons on benchmarks $f_{a_{19}}$-$f_{a_{28}}$. are given here.

Table 4
Our algorithm obtains the best or tie best in comparison with a given algorithm on 30D optimization under $f_{a_1} - f_{a_{28}}$ of our test suite.

Algorithm	Benchmarks on which our algorithm wins or obtains similar performance
iwPSO	$f_{a_1}, f_{a_3} - f_{a_5}, f_{a_7} - f_{a_{13}}, f_{a_{15}}, f_{a_{17}} - f_{a_{21}}, f_{a_{24}} - f_{a_{28}}$
CLPSO	$f_{a_1} - f_{a_5}, f_{a_7} - f_{a_{10}}, f_{a_{12}}, f_{a_{13}}, f_{a_{15}}, f_{a_{16}}, f_{a_{18}} - f_{a_{21}}, f_{a_{24}} - f_{a_{26}}, f_{a_{28}}$
DNLPSO	$f_{a_1} - f_{a_7}, f_{a_9} - f_{a_{28}}$
SLPSO	$f_{a_1}, f_{a_4}, f_{a_5}, f_{a_8}, f_{a_{10}}, f_{a_{12}}, f_{a_{13}}, f_{a_{15}}, f_{a_{17}} - f_{a_{21}}, f_{a_{25}} - f_{a_{28}}$
EPSO	$f_{a_1}, f_{a_3}, f_{a_5}, f_{a_8}, f_{a_9}, f_{a_{11}} - f_{a_{13}}, f_{a_{15}}, f_{a_{17}} - f_{a_{21}}, f_{a_{24}} - f_{a_{28}}$
MPSO	$f_{a_1}, f_{a_3} - f_{a_5}, f_{a_7} - f_{a_{10}}, f_{a_{12}} - f_{a_{21}}, f_{a_{25}}, f_{a_{26}}, f_{a_{28}}$

Table 5
A certain algorithm obtains the best or tier best performance on 30D optimization under $f_{a_1} - f_{a_{28}}$ of our test suite.

Algorithm	Benchmark on which a certain algorithm wins or obtains similar performance	Total
iwPSO	$f_{a_{21}}$	1
CLPSO	$f_{a_{14}}, f_{a_{17}}, f_{a_{19}}, f_{a_{27}}$	4
DNLPSO	f_{a_8}	1
SLPSO	$f_{a_3}, f_{a_7}, f_{a_9}, f_{a_{16}}, f_{a_{22}}, f_{a_{23}}, f_{a_{24}}$	7
EPSO	$f_{a_4}, f_{a_6}, f_{a_{10}}, f_{a_{11}}$	4
MPSO	f_{a_2}	1
Our algorithm	$f_{a_1}, f_{a_5}, f_{a_{12}}, f_{a_{13}}, f_{a_{15}}, f_{a_{18}}, f_{a_{20}}, f_{a_{21}}, f_{a_{25}}, f_{a_{26}}, f_{a_{28}}$	11

Table 6
Experiment results of the PSO-sono algorithm without fully-informed search and the PSO-sono algorithm with fully-informed search (the default) on 30D optimization under f_{a_1} - $f_{a_{28}}$ of our test suite for single-objective numerical optimization.

30D NO.	PSO-sono without fully-informed search Mean/Std.	PSO-sono with fully-informed search Mean/Std.
f_{a_1}	*0/0*(≈)	*0/0*
f_{a_2}	9.317E+06/6.956E+06(<)	**7.723E+06/8.079E+06**
f_{a_3}	1.260E+08/4.417E+08(<)	**8.586E+07/1.318E+08**
f_{a_4}	3.596E+02/3.833E+02(<)	**1.628E+02/1.250E+02**
f_{a_5}	3.790E-14/8.708E-14(<)	**2.006E-14/5.431E-14**
f_{a_6}	**7.401E+01/3.303E+01**(>)	7.587E+01/3.384E+01
f_{a_7}	3.965E+01/1.915E+01(<)	**3.298E+01/1.531E+01**
f_{a_8}	2.095E+01/5.601E-02(<)	**2.094E+01/6.063E-02**
f_{a_9}	**2.654E+01/3.619E+00**(>)	2.701E+01/3.773E+00
$f_{a_{10}}$	8.791E-02/9.356E-02(<)	**7.415E-02/6.253E-02**
$f_{a_{11}}$	2.866E+01/9.302E+00(<)	**2.784E+01/8.220E+00**
$f_{a_{12}}$	**3.443E+01/1.237E+01**(>)	3.594E+01/1.441E+01
$f_{a_{13}}$	9.671E+01/3.001E+01(<)	**9.007E+01/2.792E+01**
$f_{a_{14}}$	**2.634E+03/5.344E+02**(>)	2.663E+03/4.813E+02
$f_{a_{15}}$	3.786E+03/7.307E+02(<)	**3.534E+03/7.715E+02**
$f_{a_{16}}$	1.659E+00/5.452E-01(<)	**1.566E+00/5.609E-01**
$f_{a_{17}}$	6.234E+01/1.067E+01(<)	**5.912E+01/1.091E+01**
$f_{a_{18}}$	**5.952E+01/1.025E+01**(>)	5.954E+01/1.017E+01
$f_{a_{19}}$	3.546E+00/8.881E-01(<)	**3.267E+00/9.226E-01**
$f_{a_{20}}$	**1.081E+01/6.658E-01**(>)	1.096E+01/5.286E-01
$f_{a_{21}}$	**3.116E+02/7.051E+01**(>)	3.147E+02/6.227E+01
$f_{a_{22}}$	2.819E+03/5.970E+02(<)	**2.789E+03/5.561E+02**
$f_{a_{23}}$	3.773E+03/6.937E+02(<)	**3.644E+03/7.307E+02**
$f_{a_{24}}$	2.629E+02/1.347E+01(<)	**2.611E+02/1.304E+01**
$f_{a_{25}}$	2.856E+02/9.964E+00(<)	**2.838E+02/8.819E+00**
$f_{a_{26}}$	**2.901E+02/7.966E+01**(>)	3.047E+02/7.835E+01
$f_{a_{27}}$	8.911E+02/1.312E+02(<)	**8.757E+02/1.347E+02**
$f_{a_{28}}$	**3.156E+02/1.715E+02**(>)	3.717E+02/2.904E+02
>/≈/<	9/1/18	-/-/-

Table 7
The time complexity comparison on benchmark $f_{a_{14}}$ of our test suite for real-parameter single-objective optimization. The comparison is conducted according to the suggestion of CEC2013 competition.

Algorithms	T_0	$\widehat{T}_1$	$\widehat{T}_2$	$\frac{\widehat{T}_2-\widehat{T}_1}{T_0}$
iwPSO	0.0844	0.6428	0.7734	1.5470
CLPSO			2.9853	27.7458
DNLPSO			2.9287	27.0751
SLPSO			1.7048	12.5792
EPSO			8.8489	97.1961
MPSO			2.3534	20.2614
TSLPSO			1.3370	8.2225
PPSO			1.6729	12.2012
Our algorithm			2.8293	25.8975

CRediT authorship contribution statement

Zhenyu Meng: Conceptualization, Methodology, Supervision, Writing - review & editing, Software. **Yuxin Zhong:** Writing - original draft. **Guojun Mao:** Supervision. **Yan Liang:** Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is supported by the National Natural Science Foundation of China with the Grant No. 61906042 & No. 61773415 & No. 42077395, Natural Science Foundation of Fujian Province with the Grant No. 2021J05227 and Scientific Research Startup Foundation of Fujian University of Technology (GY-Z19013).

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.ins.2021.11.076.

References

[1] R. Eberhart, J. Kennedy, Particle swarm optimization, in: Proceedings of the IEEE international conference on neural networks, Vol. 4, IEEE, 1995, pp. 1942–1948..
[2] M. Clerc, J. Kennedy, The particle swarm – explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (1) (2002) 58–73.
[3] M.R. Bonyadi, Z. Michalewicz, Particle swarm optimization for single objective continuous space problems: a review, Evolutionary Computation 25 (1) (2017) 1–54.
[4] Z. Meng, J.-S. Pan, Monkey King Evolution: a new memetic evolutionary algorithm and its application in vehicle fuel consumption optimization, Knowledge-Based Systems 97 (2016) 144–157.
[5] J. Kennedy, The particle swarm: social adaptation of knowledge, in: Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97), Vol. 4, IEEE, 1997, pp. 303–308..
[6] Y. Shi, Particle swarm optimization: developments, applications and resources, in: Proceedings of the 2001 congress on evolutionary computation, Vol. 1, IEEE, 2001, pp. 81–86..
[7] D. Wang, D. Tan, L. Liu, Particle swarm optimization algorithm: an overview, Soft Computing 22 (2) (2018) 387–408.
[8] E.H. Houssein, A.G. Gad, K. Hussain, P.N. Suganthan, Major advances in particle swarm optimization: Theory, analysis, and application, Swarm and Evolutionary Computation 63 (2021) 100868.
[9] J. Liang, B. Qu, P. Suganthan, A.G. Hernández-Díaz, Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report 201212 (34) (2013) 281–295.
[10] T. Liao, D. Molina, M.A. Montes de Oca, T. Stützle, A note on bound constraints handling for the ieee cec05 benchmark function suite, Evolutionary Computation 22 (2) (2014) 351–359.
[11] R.C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Proceedings of the 2000 congress on evolutionary computation, Vol. 1, IEEE, 2000, pp. 84–88..
[12] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Transactions on Evolutionary Computation 10 (3) (2006) 281–295.
[13] M. Nasir, S. Das, D. Maity, S. Sengupta, U. Halder, P.N. Suganthan, A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization, Information Sciences 209 (2012) 16–36.
[14] R. Cheng, Y. Jin, A competitive swarm optimizer for large scale optimization, IEEE transactions on cybernetics 45 (2) (2014) 191–204.
[15] R. Cheng, Y. Jin, A social learning particle swarm optimization algorithm for scalable optimization, Information Sciences 291 (2015) 43–60.
[16] H. Yu, Y. Tan, J. Zeng, C. Sun, Y. Jin, Surrogate-assisted hierarchical particle swarm optimization, Information Sciences 454 (2018) 59–72.

[17] P.N. Suganthan, Particle swarm optimiser with neighbourhood operator, in: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99, Vol. 3, IEEE, 1999, pp. 1958–1962..
[18] J. Kennedy, R. Mendes, Population structure and particle swarm performance, in: Proceedings of the 2002 Congress on Evolutionary Computation, Vol. 2, IEEE, 2002, pp. 1671–1676..
[19] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Transactions on evolutionary computation 8 (3) (2004) 204–210.
[20] M.R. Bonyadi, Z. Michalewicz, Analysis of stability, local convergence, and transformation sensitivity of a variant of the particle swarm optimization algorithm, IEEE Transactions on evolutionary computation 20 (3) (2015) 370–385.
[21] Z. Meng, J.-S. Pan, K.-K. Tseng, PaDE: An enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization, Knowledge-Based Systems 168 (2019) 80–99.
[22] Z. Meng, J.-S. Pan, HARD-DE: Hierarchical archive based mutation strategy with depth information of evolution for the enhancement of differential evolution on numerical optimization, IEEE Access 7 (2019) 12832–12854.
[23] Z. Meng, Y. Zhong, C. Yang, CS-DE: Cooperative strategy based differential evolution with population diversity enhancement, Information Sciences 577 (2021) 663–696.
[24] Z. Meng, C. Yang, Hip-DE: Historical population based mutation strategy in differential evolution with parameter adaptive mechanism, Information Sciences 562 (2021) 44–77.
[25] L. Cagnina, M. Errecalde, D. Ingaramo, P. Rosso, An efficient particle swarm optimization approach to cluster short texts, Information Sciences 265 (2014) 36–49.
[26] F.E.F. Junior, G.G. Yen, Particle swarm optimization of deep neural networks architectures for image classification, Swarm and evolutionary computation 49 (2019) 62–74.
[27] J. Tian, C. Sun, Y. Tan, J. Zeng, Granularity-based surrogate-assisted particle swarm optimization for high-dimensional expensive optimization, Knowledge-Based Systems 187 (2020) 104815.
[28] A. Lin, W. Sun, H. Yu, G. Wu, H. Tang, Global genetic learning particle swarm optimization with diversity enhancement by ring topology, Swarm and evolutionary computation 44 (2019) 571–583.
[29] T.-Y. Wu, Y.-Q. Lee, C.-M. Chen, Y. Tian, N.A. Al-Nabhan, An enhanced pairing-based authentication scheme for smart grid communications, Journal of Ambient Intelligence and Humanized Computing (2021) 1–13.
[30] T.-Y. Wu, L. Yang, Z. Lee, C.-M. Chen, J.-S. Pan, S. Islam, Improved ecc-based three-factor multiserver authentication scheme, Security and Communication, Networks (2021).
[31] N. Lynn, M.Z. Ali, P.N. Suganthan, Population topologies for particle swarm optimization and differential evolution, Swarm and evolutionary computation 39 (2018) 24–35.
[32] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: 1998 IEEE International Conference on Evolutionary Computation Proceedings, Vol. 3, IEEE, 1998, pp. 69–73..
[33] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Proceedings of the 1999 congress on evolutionary computation-CEC99, Vol. 3, IEEE, 1999, pp. 1945–1950..
[34] D. Bratton, J. Kennedy, Defining a standard for particle swarm optimization, in: 2007 IEEE swarm intelligence symposium, Vol. 1, IEEE, 2007, pp. 120–127..
[35] Y. Del Valle, G.K. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, R.G. Harley, Particle swarm optimization: basic concepts, variants and applications in power systems, IEEE transactions on evolutionary computation 12 (2) (2008) 171–195.
[36] J.C. Bansal, P. Singh, M. Saraswat, A. Verma, S.S. Jadon, A. Abraham, Inertia weight strategies in particle swarm optimization, in: 2011 Third world congress on nature and biologically inspired computing, IEEE, 2011, pp. 633–640..
[37] M. Taherkhani, R. Safabakhsh, A novel stability-based adaptive inertia weight for particle swarm optimization, Applied Soft Computing 38 (2016) 281–295.
[38] K.R. Harrison, A.P. Engelbrecht, B.M. Ombuki-Berman, Inertia weight control strategies for particle swarm optimization, Swarm Intelligence 10 (4) (2016) 267–305.
[39] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE transactions on evolutionary computation 1 (1) (1997) 67–82.
[40] R. Cheng, Y. Jin, A social learning particle swarm optimization algorithm for scalable optimization, Information Sciences 291 (2015) 43–60.
[41] N. Lynn, P.N. Suganthan, Ensemble particle swarm optimizer, Applied Soft Computing 55 (2017) 533–548.
[42] H. Liu, X. Zhang, L. Tu, A modified particle swarm optimization using adaptive strategy, Expert Systems With Applications 152 (2020) 113353.
[43] T. Peram, K. Veeramachaneni, C.K. Mohan, Fitness-distance-ratio based particle swarm optimization, in, in: Proceedings of the 2003 IEEE Swarm Intelligence Symposium, IEEE, 2003, pp. 174–181.
[44] A. Ratnaweera, S.K. Halgamuge, H.C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Transactions on evolutionary computation 8 (3) (2004) 240–255.
[45] B.-Y. Qu, P.N. Suganthan, S. Das, A distance-based locally informed particle swarm model for multimodal optimization, IEEE Transactions on evolutionary computation 17 (3) (2012) 387–402.
[46] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE transactions on Evolutionary Computation 13 (2) (2008) 398–417.
[47] M. Ghasemi, E. Akbari, A. Rahimnejad, S.E. Razavi, S. Ghavidel, L. Li, Phasor particle swarm optimization: a simple and efficient variant of pso, Soft Computing 23 (19) (2019) 9701–9718.
[48] G. Xu, Q. Cui, X. Shi, H. Ge, Z.-H. Zhan, H.P. Lee, Y. Liang, R. Tai, C. Wu, Particle swarm optimization based on dimensional learning strategy, Swarm and Evolutionary Computation 45 (2019) 33–51.
[49] Z. Meng, J.-S. Pan, H. Xu, Quasi-affine transformation evolutionary (quatre) algorithm: a cooperative swarm based algorithm for global optimization, Knowledge-Based Systems 109 (2016) 104–121.
[50] Z. Meng, J. Pan, QUasi-Affine TRansformation Evolution with External ARchive (QUATRE-EAR): An enhanced structure for differential evolution, Knowledge Based Systems 155 (2018) 35–53.