Swarm and Evolutionary Computation 78 (2023) 101274

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Reinforcement learning-based particle swarm optimization with neighborhood differential mutation strategy

Wei Li [a], Peng Liang [a], Bo Sun [a], Yafeng Sun [a], Ying Huang [b,*]

[a] School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China
[b] School of Mathematical and Computer Sciences, Gannan Normal University, Ganzhou 341000, China

ARTICLE INFO

Keywords:
Particle swarm optimization
Nonlinear inertia weight
Reinforcement learning
Differential mutation
Cosine similarity

ABSTRACT

The particle swarm optimization (PSO) algorithm has been one of the most effective methods for solving various engineering optimization problems. Most existing PSO variants frequently use fixed operators, the adoption of a fixed operator learning mode may restrict the intelligence level of each particle, thus reducing the performance of PSO in solving optimization issues with complicated fitness landscapes. To address single goal real-parameter numerical optimization while overcoming the above shortcoming, this paper proposes a reinforcement learning-based particle swarm optimization with neighborhood differential mutation strategy (NRLPSO). In NRLPSO, a dynamic oscillation inertial weight (DOW) strategy that provides particles with dynamic adjustment ability in different situations is designed. To resolve the operator selection conundrum of exploration and exploitation, a reinforcement learning-based velocity vector generation (VRL) strategy is developed. At each iteration, particles select the velocity update model based on reinforcement learning, and VRL helps to thoroughly search the problem space. A velocity updating mechanism based on cosine similarity (VCS) is applied to control the velocity learning mode to determine more promising solutions. Furthermore, to alleviate the problem of premature convergence, a local update strategy with neighborhood differential mutation (NDM) is employed to increase the diversity of the algorithm. To verify the efficiency of the proposed algorithm, the CEC2017 and CEC2022 test suites are implemented, and nine classic or state-of-the-art PSO variants are comprehensively tested. The experimental results show that NRLPSO outperforms the popular PSO variants in terms of convergence speed and accuracy. Since NRLPSO utilizes the DE mutations, it is compared with the representative LSHADE variant algorithm - LSHADE_SPACMA. Although LSHADE_SPACMA is better than NRLPSO concerning algorithm stability and convergence accuracy, we will refine our work in the future to enhance the performance in all aspects.

1. Introduction

In the past thirty years, the efficiency of classical heuristic algorithms has made these unsuitable for handling challenging optimization issues. Scholars have been searching for a better way to solve these issues, which has led to the development of metaheuristic algorithms, also known as intelligent optimization algorithms. Compared with traditional heuristic algorithms, such as the calculus-based method and exhaustive method, metaheuristic algorithms are mature global optimization algorithms with high robustness and wide applicability. In the past decade, metaheuristic algorithms have rapidly developed, with obvious application prospects in various research fields, such as medical treatment, engineering, and telecommunications [1–5]. Metaheuristic algorithms include the genetic algorithm (GA) [6], artificial bee colony (ABC) [7], differential evolution (DE) [8], simulated annealing (SA) [9], ant colony optimization (ACO) [10] and particle swarm optimization (PSO) [11]. Among these algorithms, PSO is widely employed because of its few parameters, simple structure, and easy implementation.

The most well-known metaheuristic optimization algorithm is PSO. Inspired by the imitation of social behaviors (such as bird clustering and fish swarms), the PSO algorithm was proposed by Kennedy and Eberhart in 1995 and explores the problem space by updating the velocity and position of particles [11]. However, the study revealed that PSO has the flaws of premature convergence and poor convergence performance, with an increase in the optimization problem dimension [12–14]. Hence, to alleviate the shortcomings of PSO, many PSO variants have been successively proposed by scholars, and can be divided

* Corresponding author.
E-mail addresses: liwei@jxust.edu.cn (W. Li), liangpeng@mail.jxust.edu.cn (P. Liang), sunbo@mail.jxust.edu.cn (B. Sun), sunyafeng_jx@mail.jxust.edu.cn (Y. Sun), nhwshy@whu.edu.cn (Y. Huang).

https://doi.org/10.1016/j.swevo.2023.101274
Received 7 August 2022; Received in revised form 3 February 2023; Accepted 9 February 2023
Available online 25 February 2023

into five categories: (1) Parameter tuning [15–19], (2) Hybridization [20–23], (3) Topological structure [24–29], (4) Multi population [30–35], (5) Learning strategy [36–39].

In recent years, the creation of PSO with various topologies and learning strategies has become a focus of research [40]. In general, these PSO variants are designed to increase population diversity and alleviate premature convergence problems. In particular, five famous topologies (including all, rings, clusters, and pyramids) are discussed in [25]. A fully informed PSO is introduced in [13], in which the velocity of each particle is updated according to the information of its neighborhoods. In Ref. [27], a locally informed PSO is also proposed to guide particles to explore the problem space by selecting several personal optimal positions in the neighborhood. In addition, the personal best *Pbest* and the global best *Gbest* are updated utilizing distance-based dynamic neighborhood to enhance the capabilities of escaping from local optimal in [29]. And the hybridization optimization method, which combines PSO with DE, has attracted increasing attention [41,42]. In particular, a locally evolved PSO algorithm is developed in [43] to improve the locally optimal particle search in PSO using DE operations. This algorithm fully utilizes the advantages of the standard DE and improves the ability of PSO to identify the global optimal particle.

Many mutation operators have been designed to address a variety of challenging optimization issues. Although most existing PSO variants frequently use fixed operators, the search for the optimal solution is a complex process. The adoption of a fixed operator learning mode may restrict the intelligence level of each particle, thus reducing the performance of PSO in solving optimization problems with complicated fitness landscapes. Hence, the adaptive selection of optimal operators throughout the search process has been the subject of some studies in recent years. This paper provides a reinforcement learning based operator selection strategy to address the exploration and exploitation conundrum in operator selection. Reinforcement learning has been used to control state transition, and the method identifies the appropriate operator for each parent that maximizes its cumulative improvement. This method divides the state of population search into four categories and selects the opportune operator varying with the change in population state. The contributions of this paper have the following aspects:

(1) A dynamic oscillation inertial weight strategy is proposed to expand or restrict the search process to more effectively balance exploration and exploitation of the algorithm.
(2) The evolution process of the population is divided into four search states, and appropriate search strategies are dynamically assigned according to these states. Adaptive adjustment of particle states through proposed a reinforcement learning-based velocity vector generation strategy.
(3) A velocity updating mechanism based on cosine similarity is proposed. Particles can better learn towards more promising solutions in complex problem spaces by controlling the learning paradigms of particle velocity.
(4) A local update strategy with neighborhood differential mutation is proposed to update the information of the *Pbest* and *Gbest* neighborhoods, which helps the algorithm jump out of the local optima situation.

The remainder of this paper is organized as follows: Section 2 introduces classic PSO, reinforcement learning, and PSO with the dynamic neighborhood. In addition, the concept of cosine similarity is introduced. Section 3 introduces a reinforcement learning-based particle swarm optimization with neighborhood differential mutation strategy (NRLPSO). Section 4 discusses and analyzes experimental results. Section 5 gives the conclusion and directions for future work.

2. Related work

2.1. Classic PSO

PSO is a metaheuristic algorithm utilized in global random search. In the classic PSO, each particle explores the issue space by utilizing its prior knowledge and collaborating with other particles, and the potential solution of each problem to be optimized is represented as a particle in the search space. In the D-dimensional search space, the position of the ith particle at the tth iteration is denoted as $x_i(t) = (x_{i1}(t), x_{i2}(t), \ldots, x_{iD}(t))$. The velocity of particle i at the tth iteration is represented by $v_i(t) = (v_{i1}(t), v_{i2}(t), \ldots, v_{iD}(t))$. The updating equations of particle velocity and position are shown as (1) and (2).

$$v_i(t+1) = \omega v_i(t) + c_1 r_1 (Pbest_i(t) - x_i) + c_2 r_2 (Gbest(t) - x_i(t)) \tag{1}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2}$$

where $Pbest_i$ represents the historical optimal solution of the particle and *Gbest* represents the historical optimal solution of the whole population. ω is the inertial weight, which is used to control the influence of the previous velocity on the current velocity; r_1 and r_2 are two numbers randomly selected from the uniform distribution $[0, 1]$. c_1 is the individual cognitive acceleration coefficient, and c_2 is the social acceleration coefficient.

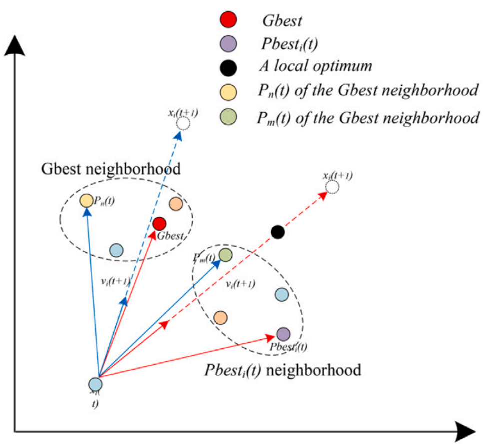


Fig. 1. Schematic diagram of particle trajectory.

2.2. Dynamic neighborhood

The classic PSO only updates each particle based on the *Pbest* of a single particle and the *Gbest* obtained by the whole particle swarm. Fig. 1 shows the shortcomings of the update strategy of the classic PSO, where the red particle represents *Gbest* and $P_n(t)$ is a particle in the *Gbest* neighborhood. The black particle indicates a local optimum. The purple particle $Pbest_i(t)$ is the individual historically optimal solution of particle $x_i(t)$, and $P_m(t)$ is a particle in the $Pbest_i(t)$ neighborhood.

The addition of vectors in Fig. 1 satisfies the parallelogram law. Due to the influence of acceleration coefficients c_1 and c_2 and random numbers r_1 and r_2, the step size of particle movement will slightly change. The red dotted line is the direction in which particle $x_i(t)$ will move after learning from paradigms *Pbest* and *Gbest*. There is a locally optimal solution in the direction of the red dotted line. When the particle moves along the track of the red dotted line, it may fall into the local optimal solution. The blue dotted line is the direction in which particle $x_i(t)$ will move after learning from paradigms $P_n(t)$ and $P_m(t)$. When the particle moves along the blue dotted line, it avoids the local optimum. Hence, this paper draws the conclusion that fully utilizing the information of the swarm could alleviate the premature convergence problem, especially for functions with many locally optimal solutions.

To fully utilize the neighborhood information of other particles, the dynamic neighborhood is selected to enhance information communication in the population. The neighborhood of particle $Pbest_i$ is determined to be the closest k particles by computing the Euclidean distance between all particles in the current population and all particles in *Pbest*. By calculating the Euclidean distance between all particles in the population and all particles in *Gbest*, the neighborhood of particle *Gbest* is defined as the nearest k particles. Each round of evolution automatically adjusts the neighborhood of *Pbest* and *Gbest*.

2.3. Reinforcement learning

Reinforcement learning is a crucial machine learning technique that enables an agent to discover the mapping relationship between states and behaviors, the continual interaction with the environment through trial and error to maximize long-term cumulative returns [44]. At present, reinforcement learning methods have been widely used in intelligent control, robotics, and other fields [45–47]. The principal elements of reinforcement learning include learning agents, environments, states, actions, and rewards. Reinforcement learning aims to obtain the optimal solution for the finite or infinite state of a number of dynamic problems, which is related to how an agent should act in a dynamic environment [44]. To realize the combination of reinforcement learning and PSO in this study, the idea of the Q-learning algorithm is applied [48].

In reinforcement learning, different state–action pairs are represented by different Q value. Standard model-free reinforcement learning techniques utilized in this paper is Q-learning [49–53]. In Q-learning, the Q value is updated as (3).

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a Q_t(s_{t+1}, \alpha) - Q_t(s_t, a_t)] \tag{3}$$

where s_t and a_t represent the current state and the current behavior, respectively; s_{t+1} represents the next state; r_{t+1} is the immediate reward acquired from executing action a_t; γ is the discount factor within [0,1]; α is the learning rate within [0,1]; and $Q_t(s_t, a_t)$ is the total cumulative reward that the learning agent has gained.

Reinforcement learning has been employed in PSO and other evolutionary algorithms. Reinforcement learning is used in the parameter adjustment of the differential evolution algorithm in [54]. Piperagkas et al. proposed the integration of reinforcement learning and PSO [55]. Reinforcement learning works independently of PSO in [56], and is applied to enhance the estimation of objective functions in noise problems. In some recent studies, a new memetic PSO based on reinforcement learning to solve the problem of when and how often local search methods should be employed, and which particle should perform local search operations is introduced [57]. A decomposition-based multiobjective evolutionary algorithm reinforcement learning auxiliary parameter tuning is used to control the parameters of the evolutionary algorithm in [58]. A reinforcement learning-based communication topology in PSO is proposed to control the dynamic topology [59]. By application of reinforcement learning in the DE algorithm, Tian Y et al. proposed an adaptive operator selection evolutionary multiobjective optimization algorithm based on deep reinforcement learning that solves the dilemma of exploration and exploitation in operator selection [60].

2.4. Cosine similarity

Cosine similarity is applied to evaluate the similarity of two vectors by calculating the cosine of their included angle. Angles of 0 degrees have a cosine of 1, whereas angles of any other degrees have a cosine that is not larger than 1; its minimum value is negative 1. Whether or not the two vectors point substantially in the same direction is determined by the cosine of the angle between them. Two vectors that point in the same direction have a cosine similarity value of 1. Cosine similarity is 0 for a 90 degrees angle between two vectors. When two vectors are pointing in opposite directions, the cosine similarity equals

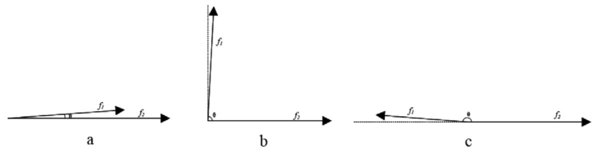
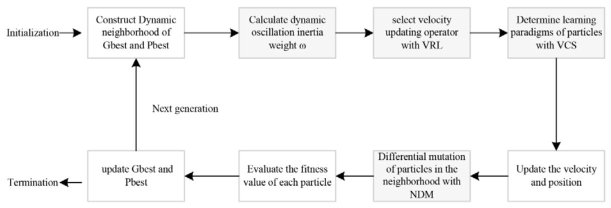


Fig. 2. Cosine similarity.



Initialization
Construct Dynamic neighborhood of Gbest and Pbest
Calculate dynamic oscillation inertia weight ω
select velocity updating operator with VRL
Determine learning paradigms of particles with VCS
Next generation
Update the velocity and position
Differential mutation of particles in the neighborhood with NDM
Evaluate the fitness value of each particle
update Gbest and Pbest
Termination

Fig. 3. General procedure of NRLPSO.

−1. Cosine similarity does not depend on the length of the vector and only depends on the direction in which the vector is pointing. Cosine similarity is usually applied to positive spaces, so the value given falls in [−1,1], which is the most common 2-D space, as shown in Fig. 2.

As shown in Fig. 2(a), the closer the cosine value is to 1, the closer the included angle is to 0 degrees, that is, the more similar the directions of the two vectors. As shown in Fig. 2(b), the closer the cosine value is to 0, the closer the two vectors are to being orthogonal, that is, the closer the two vectors are to being independent. As shown in Fig. 2(c), the two vectors are more nearly pointing in opposing directions the closer the cosine value is to −1.

The cosine similarity does not depend on the length of the vector and only depends on the direction in which the vector is pointing. The calculation equation for cosine similarity is expressed as (4).

$$\cos(\theta) = \frac{\sum_{i=1}^{n}\left(A_i * B_i\right)}{\sqrt{\sum_{i=1}^{n}(A_i)^2} * \sqrt{\sum_{i=1}^{n}(B_i)^2}} \tag{4}$$

where A_i and B_i are the ith components of A and B respectively.

Some recent studies have explored cosine similarity. The characteristics of the CNN bus are analyzed through cosine similarity in [61]. A protection decision based on the cosine similarity index (CSI) to measure the alignment of two data windows at both ends of a line segment is proposed in [62]. The cosine similarity measure is applied to the setting of spherical fuzzy sets in [63].

3. Proposed algorithm

The general procedure of NRLPSO is shown in Fig. 3. The innovation of the proposed algorithm lies in the following findings: (1) A new dynamic oscillation inertia weight that can make particles better adapt to the problem space and has better dynamic adjustment ability is proposed. (2) A reinforcement learning-based velocity vector generation strategy is proposed to control the state of particles, and particles can switch the strategy to be executed according to the state. The particle performs the most suitable search strategy based on its state to improve the optimization ability of the algorithm. (3) A velocity updating mechanism based on cosine similarity is introduced to analyze information similarity between two learning paradigms, and a velocity updating mechanism is designed based on cosine similarity. (4) The dynamic neighborhood is adopted in this paper to fully utilize information about the population. On this basis, a local update strategy with neighborhood differential mutation strategy is proposed to enhance the flow of information within the neighborhood. The effective algorithm can alleviate the situation in which the algorithm prematurely falls into a local optimum.

In the proposed NRLPSO, particles in the population are divided into four states, VRL is used to control the transfer of particle states, and particles are assigned appropriate search methods according to various states. After the particle performs the corresponding search methods, it obtains the corresponding reward, which is determined by the fitness improvement and evolutionary factor [23]. The calculation method of the reward is shown as (5).

$$R = \begin{cases} 2, if \quad f(x_{new}) < f(x_{old}) \quad and \quad E_f(x_{new}) > E_f(x_{old}) \\ 1, if \quad f(x_{new}) < f(x_{old}) \quad and \quad E_f(x_{new}) \le E_f(x_{old}) \\ 0, if \quad f(x_{new}) \ge f(x_{old}) \quad and \quad E_f(x_{new}) > E_f(x_{old}) \\ -2, if \quad f(x_{new}) \ge f(x_{old}) \quad and \quad E_f(x_{new}) \le E_f(x_{old}) \end{cases} \tag{5}$$

where E_f is the evolutionary factor, which represents population diversity, and the larger the E_f the lower the population diversity. The evolutionary factor of this study is calculated as (6).

$$E_f(i) = \frac{d_i - d_{min}}{d_{max} - d_{min}} \tag{6}$$

where d_{max} and d_{min} indicate the longest and shortest distances among d_i, respectively. d_i is the mean distance between the ith particle and other particles, which is calculated as (7).

$$d_i = \frac{1}{N-1} \sum_{j=1, j\ne i}^{N} \sqrt{\sum_{k=1}^{D} (x_i^k - x_j^k)^2} \tag{7}$$

where N is the population size and D is the dimension of the problem space.

3.1. Dynamic oscillation inertia weight

As one of the parameters of PSO, inertia weight (ω) enables dynamic adjustment ability for particles in various situations, to achieve the balance between exploration and exploitation. In general, in population-based optimization techniques, a fairly high degree of diversity is required in the early stages of the search to allow the entire search problem space. To successfully acquire the optimal solution fine-tuning the solution is crucial in the late stages of the search. Hence, inertia weight has an important role in PSO. Classic PSO adopts inertia weight with a fixed value [11], so particles cannot make adaptive adjustments according to different environments, which causes the algorithm to easily fall into local optimal. Shi Y et al. discovered that the performance of PSO significantly improved with a linear change in inertia weight [18]. Some studies have adopted linear adaptive weights [2,59,64,65]. However, for complex optimization problems, nonlinear adaptive weights can better fit the environment and have a better dynamic adjustment ability [16,66,67]. In this paper, a new nonlinear inertia weight is determined by (8) and (9).

$$r(t+1) = 4r(t)(1-r(t)) \qquad r(0) = rand(1) \tag{8}$$

$$\begin{aligned} \omega(t) =& u - ((\frac{FEs}{MaxFEs} - \omega_{max}) * r * \omega_{min} + v * (\omega_{max} \\ & - \omega_{min}) * (\frac{FEs}{MaxFEs})) \end{aligned} \tag{9}$$

where $r(0)$ is a random number distributed in [0,1]; $\omega_{max} = 1$ and $\omega_{min} = 0.4$; FEs represents the number of current evaluations; $MaxFEs$ represents the maximum number of evaluations; $u = 0.6$ and $v = 0.33$. The corresponding simulation diagram is shown in Fig. 4.

As shown in Fig. 4, the inertia weight ω is a nonlinear form that presents a downwards trend, which also fits the swarm search process. ω will shock in the iterative process, and the oscillation amplitude gradually decreases with evolution. At the beginning of the iterative process, the movements of the particles are more diversified due to larger amplitudes, enhancing the diversity of the population. At the late stage of the iterative process, smaller amplitudes are favorable for particle convergence. This mode of nonlinear inertial weight enables particles to have better fitting and simulation capabilities, and can better balance the global exploration and local exploitation ability of particles. In addition, the wave properties of inertia weight make particles move more randomly so that particles can avoid falling into the local optimum as much as possible.

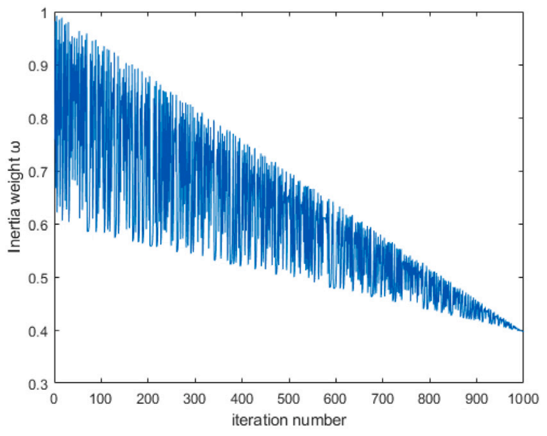


Fig. 4. Dynamic oscillation inertia weight.

3.2. Reinforcement learning-based velocity vector generation strategy

This section will introduce how to integrate reinforcement learning with PSO. In the NRLPSO, the fundamental elements of reinforcement learning comprise learning agents, environments, states, actions, and rewards. In this study, the environment is the search space for particles. Particle states are classified as exploration, exploitation, convergence, and jumping out, each particle in the population is in one of these states. The definition of an action is the transition from one state to another, and the action can be selected according to the Q-Table by performing this action to achieve the target state [68].

The Q value is updated as (3). One of the key parameters of Q-learning is how the learning rate α controls the degree to which the newly learned information replaces previously held knowledge. For instance, Q-learning undertakes more exploration for all defined states when α is close to 1, assigning the newly acquired information a higher priority. A small α value gives a higher priority for the existing information in the Q-Table to be exploited. Hence, α is typically set at a high value at the start of the search process and gradually decreased, which is calculated as (10).

$$\alpha(t) = 1 - (0.9 * \frac{FEs}{MaxFEs}) \tag{10}$$

where FEs represents the number of current evaluations and $MaxFEs$ represents the maximum number of evaluations.

Fig. 5 shows the overall NRLPSO structure integrating reinforcement learning and PSO. In the classic PSO, the search operation starts at the beginning of the entire particle swarm. At the end of the search process, the operation gradually switches to convergence. In the process of NRLPSO search, particles are adaptively switched between states through reinforcement learning based on their current states. The particle selects the most appropriate search operation according to its state. After an operation, positive incentives are offered to particles that perform well, and punishments are given to particles that perform poorly. The rewards are calculated as (9). The process is repeated up until the point where the FES maximum was reached.

Particle states are determined through reinforcement learning; the paradigm of particle learning based on state selection is shown in Table 1. More details are provided in [59].

According to (1), large values of c_1 favor individual searches, while small values of c_2 lessen the impact of $Gbest$. Exploring the search space completely and getting the particles out of local optimal are the main objectives of the exploitation and exploration states. In this case, c_2 has a tiny value but c_1 has a large value in both the exploitation and exploration states. In the jumping-out state, the particles are pushed away from their existing optimal positions by using a large value for c_2 and a small value for c_1.

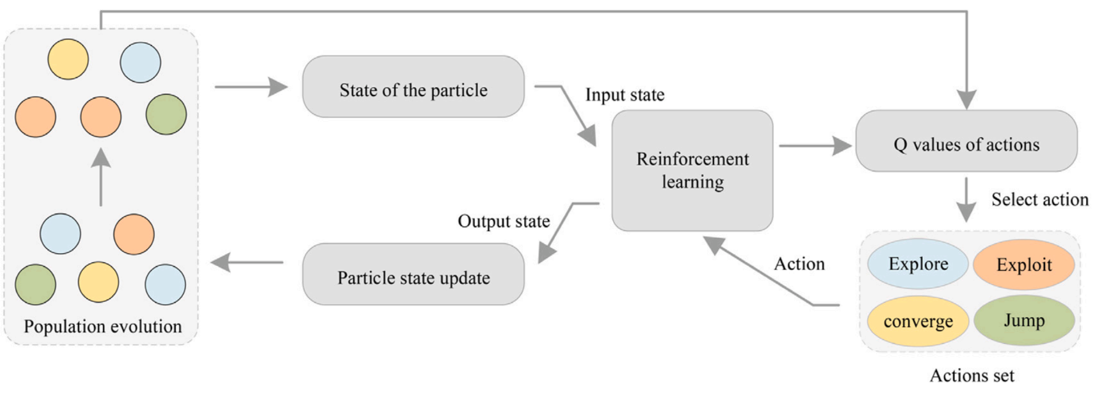


Fig. 5. Reinforcement learning-based velocity vector generation strategy.

Table 1
Particle learning paradigms and acceleration coefficients.

Particle state	c_1	c_2	$Paradigm_1$	$Paradigm_2$
Exploration	2.2	1.8	$Pbest_i$ P_b	P_a
Exploitation	2.1	1.9	P_b	$Gbest$ P_a
Convergence	2	2	$Pbest_i$	$Gbest$ $Gbest$
Jumping-out	1.8	2.2	P_b	P_a

With the exception of value settings for acceleration coefficients c_1 and c_2, the following is a more thorough explanation of the four searching states:

(1) In the exploration state, all particles are as far as possible to avoid falling into the local optimal solution. When the cosine similarity of the two learning paradigms is less than 0, a particle named P_a is randomly selected to replace the current $Gbest$ in the distance-based $Gbest$ dynamic neighborhood. Otherwise, particle x_i explores around $Pbest_i$. In this way, particles can have a wider search area, rather than crowding around the current $Gbest$ and falling into local optimal.
(2) In the exploitation state, searching the issue space as thoroughly as possible is requested for particles, particularly the areas near $Pbest_i$ of each particle. When the cosine similarity of the two learning paradigms is less than 0, a particle named P_b is randomly selected to replace the current $Pbest_i$ in the distance-based $Pbest_i$ dynamic neighborhood. Otherwise, particle x_i exploits around $Gbest$. In this way, particles utilize the information near $Gbest$ to obtain better feasible solutions.
(3) In the convergence state, when the cosine similarity of the two learning paradigms is less than 0, the NRLPSO applies the same velocity update method as the classic PSO. Otherwise, the particle learns from the $Gbest$ position.
(4) In the jumping-out state, the goal of every particle is to develop a better solution by escaping from the local optimal. In this case, particle P_a is randomly selected from the $Gbest$ neighborhood and particle P_b is randomly selected from the $Pbest$ neighborhood. The velocity update equation is as shown in (11).

$$v_i(t+1) = \omega v_i(t) + c_1 * rand(1,D). * (P_b - x_i(t)) + c_2 * rand(1,D). * (P_a - x_i(t)) \tag{11}$$

where t represents the current iteration number; ω is the inertia weight; D is the dimension of the problem space; P_a and P_b are particles randomly selected from the $Gbest$ neighborhood and $Pbest_i$ neighborhood, respectively; c_1 and c_2 are acceleration coefficients.

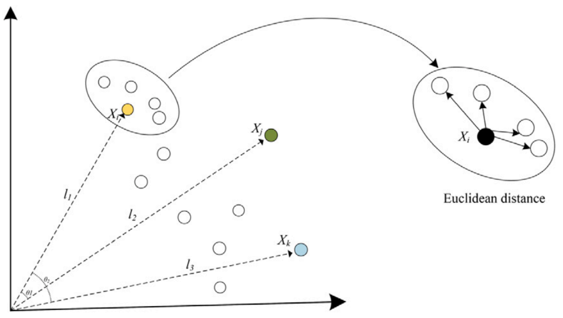


Fig. 6. Euclidean distance and cosine similarity to judge similarity.

3.3. Velocity updating mechanism based on cosine similarity

In the classic PSO velocity update equation as (1), particles obtain information from two paradigms $Pbest$ and $Gbest$. The selection of paradigms has a crucial role in guiding particle movement; Good paradigms can better guide particle movement, and bad paradigms may cause the particles to premature particles becoming local optimal. Most scholars focus on how to choose good learning paradigms. However, few scholars pay attention to the influence of the connections between two paradigms on particle movement. Cosine similarity is employed in this study to analyze the similarity between two paradigms, and a velocity updating mechanism based on cosine similarity is designed for particle velocity updating.

In high-dimensional space, it is common to judge the similarity between two vectors based on Euclidean distance. The greater the distance is, the greater the difference between two individuals. The angle between two vectors in vector space are used in cosine similarity to measure the differences between two individuals. In general, while cosine similarity depicts the relative difference in direction, the Euclidean distance depicts the absolute difference in value. In [69], the two ways of judging similarity are explored in depth. In this study, it is considered that in a complex high-dimensional space, it is better to judge the difference between vectors by the difference in direction. Fig. 6 shows the different forms of using Euclidean distance and cosine similarity to judge the difference between two vectors in 2-dimensional target space. The smaller the Euclidean distance of particle x_i, the more similar the particle is considered. For cosine similarity to judge the similarity between two particles, θ_1 is less than θ_2, so x_i is more similar to x_j.

Where x_i, x_j and x_k are the particles distributed in the 2-dimensional target space; l_1, l_2 and l_3 are the corresponding vectors; θ_1 is the angle between the vectors of x_i and x_j; and θ_2 is the angle between the vectors x_i and x_k.

Algorithm 1. Velocity updating mechanism based on cosine similarity

```
1. Population $N$; Dimension $D$; /*initialization*/
2. Velocity update parameters: $\omega$, $c_1$, $c_2$, $r_1$, $r_2$;
3. Randomly initialize position vector $x_i$, velocity vector $v_i$;
4. While termination condition is not satisfied do
5.    Select learning paradigms $pos_1$ and $pos_2$ of particles as Table 1;
6.    Use (4) to calculate the similarity between paradigms, $CS$;
7.    /*Particle velocity update*/
8.    If ($CS < 0$)
9.       For $i = 1 : N$ do
10.         $v_i = \omega * v_i + c_1 * r_1 * (x_i - pos_1) + c_2 * r_2 * (x_j - pos_2)$
11.      End
12.   Else
13.      For $i = 1 : N$ do
14.         As shown in Table 1, particles learn from a paradigm;
15.      End
16.   End
17. Until the maximum number of FEs is met
```

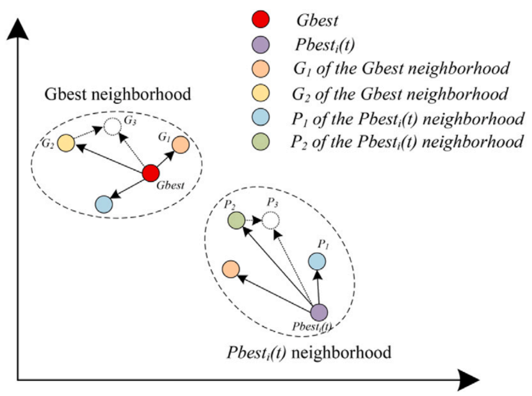

Gbest
Pbest_i(t)
G_1 of the Gbest neighborhood
G_2 of the Gbest neighborhood
P_1 of the Pbest_i(t) neighborhood
P_2 of the Pbest_i(t) neighborhood
Gbest neighborhood
G_3
G_2
G_1
Gbest
P_2
P_3
P_1
Pbest_i(t)
Pbest_i(t) neighborhood

Fig. 7. A local update scheme with neighborhood differential mutation.

In this paper, cosine similarity is employed to judge the similarity between two learning paradigms. When the similarity between two paradigms is high, the fewer feasible solutions of the objective function contained between two vectors, and the fewer information particles can learn from these paradigms, this is not conducive to particles jumping out of the local optimal solution. As shown in Fig. 6, x_i and x_j contain fewer feasible solutions. In contrast, when the similarity between the samples is low, the more feasible solutions of the objective function are contained between the two vectors, and the diversity of the population can be increased by learning from these samples, which is beneficial to global exploration. The more feasible solutions x_i and x_k contain, the more information a particle can obtain from them. Based on this idea, a velocity updating mechanism based on cosine similarity is proposed. Relevant pseudocodes are presented as follows:

Algorithm 1 is used to judge the similarity between two learning paradigms when the particle velocity is updated. Particles learn from one of the paradigms when the similarity between two paradigms is high. Particles learn from two paradigms when the diversity among paradigms is high. This strategy makes the particles move towards a more promising position every time they move.

In the pseudocode of **Algorithm 1**, the parameters are set (Lines 1 to 2). Random initialization is used to determine the position and velocity of particles (Line 3). Iteration of the algorithm begins (Line 4). The paradigms of particle learning are selected (Line 5). The cosine similarity of the two learning paradigms is calculated (Line 6). The particle velocity update mode is selected based on cosine similarity (Lines 8 to 14). Algorithm iteration terminates (Line 15).

3.4. Local update strategy with neighborhood differential mutation

The DE algorithm is a random population-based evolutionary algorithm. Recently, many studies have been focused on the effectiveness of combining the DE algorithm with the PSO algorithm [22,23]. The capacity to escape from the local optimal is improved by the hybridization of the DE algorithm and PSO algorithm, which also enhances the variety of *Pbest* and *Gbest*.

In this paper, to fully utilize the neighborhood information of other particles, the dynamic neighborhood is used to enhance the communication between neighborhoods. On this basis, this paper designs a local update scheme with neighborhood differential mutation. The diversity of the neighborhood is increased by introducing a differential mutation operator to perturb the particles in the neighborhood.

In Fig. 7, the red particle is the global historical optimal particle *Gbest*, and G_1 and G_2 are particles in the dynamic neighborhood of *Gbest*, in which G_1 is the closest to the Euclidean distance of *Gbest* and G_2 is the farthest from the Euclidean distance of *Gbest*. The purple particle is the individual historical optimal location. P_1 and P_2 are particles in the dynamic neighborhood of $Pbest_i$, in which P_1 is the closest to the Euclidean distance of $Pbest_i$ and P_2 is the farthest from the Euclidean distance of $Pbest_i$.

In a local update scheme with neighborhood differential mutation, when $Pbest_i$ is not updated for two consecutive iterations, difference variation is performed on the *Gbest* neighborhood and $Pbest_i$ neighborhood. In the *Gbest* neighborhood, The (12) is used to generate a differential mutation vector. The minimum optimization problem is selected as an example for the update in the neighborhood. If the particle fitness value obtained after the differential variation is less than *Gbest*, then *Gbest* in Fig. 7 is updated to the G_3 position. Otherwise, the G_2 position is updated to the G_3 position. In the $Pbest_i$ neighborhood, P_1 and P_2 are formed as differential vectors using the (13). If the fitness value of the mutated particle is less than $Pbest_i$, $Pbest_i$ will be updated to P_3. Otherwise, the P_2 position is updated to the P_3 position.

$$VGbest(t) = Gbest(t) + rand(1, D).*(G_1 - G_2) \tag{12}$$

$$VPbest_i(t) = Pbest_i(t) + rand(1, D).*(P_1 - P_2) \tag{13}$$

where t represents the current iteration number, and D is the dimension of the problem space. $Gbest(t)$ is the current global historical best position. G_1 and G_2 are the nearest particle to $Gbest(t)$ and furthest particle to $Gbest(t)$, respectively, in the $Gbest(t)$ neighborhood. $Pbest_i(t)$ is the current individual historical optimal position. P_1 and P_2 are the nearest particle to $Pbest_i(t)$ and furthest particle to $Pbest_i(t)$, respectively, in the $Pbest_i(t)$ neighborhood.

By means of differential variation in the neighborhood, the diversity within the neighborhood is enhanced and the particle can be effectively prevented from falling into a local optimum. The pseudocode of the proposed strategy is expressed as follows:

Algorithm 2 is used to update the neighborhood of particles and to carry out a differential mutation operation with the particles with the best performance in the neighborhood and those with the worst performance in the neighborhood. This algorithm replaces the original $f(Pbest_i)$ or $f(Gbest)$ when the mutated particle performs better. Otherwise, the worst performing particle in the neighborhood is replaced by the mutated particle.

In the pseudocode of **Algorithm 2**, the parameters are set (Lines 1 to 2). The position and velocity of the particle are randomly initialized (Line 3). Variable initialization is performed (Lines 3 to 4). Iteration of the algorithm begins (Line 5). The dynamic neighborhood (Lines 6 to 7). When the update of $Pbest_i$ stagnates, the differential variation update of particles in the neighborhood of $Pbest_i$ and *Gbest* are carried out (Lines 8 to 21). The fitness of each particle is calculated, *Gbest* and $Pbest_i$ are updated. Algorithm iteration terminates (Line 25).

Algorithm 2. A local update scheme with neighborhood differential mutation

```
1. Population $N$; Dimension $D$; Number of iterations $t$; Neighborhood size $k$;
2. Randomly initialize position vector $x_i$, velocity vector $v_i$;
3. Initialization the $Gbest$ position, initialization the $Pbest$ positon;
4. Initialization the $VGbest$ position, initialization the $VPbest$ positon;
5. While termination condition is not satisfied do
6.    Identify $Pbest_i(t)'$ neighborhood ;
7.    Identify $Gbest's$ neighborhood ;
8.    If $(t > 2)\&\&(Pbest(t) == Pbest(t-1))$
9.       $VGbest$ is calculated by (11);
10.         If $f(VGbest) < f(Gbest)$
11.            Update $Gbest$ location and $f(Gbest)$ value;
12.         Else
13.            Replace the particle farthest from $Gbest$ in the $Gbest$ neighborhood with $VGbest$;
14.         End
15.      $VPbest_i(t)$ is calculated by (12);
16.         If $f(VPbest_i(t))<f(Pbest_i)$
17.            Update $Pbest_i$ location and $f(Pbest_i)$ value;
18.         Else
19.            Replace the particle farthest from $Pbest_i$ in the $Pbest$ neighborhood with $VPbest$;
20.         End
21.   End
22. Calculate of the fitness of each particle $f_i$;
23. Update the $Gbest$ position, update the $Pbest$ position;
24. $t = t + 1$;
25. Until the maximum number of FEs is met
```

3.5. Computational complexity analysis

In this paper, the proposed algorithm contains four main components, the DOW strategy, the VRL strategy, the VCS mechanism, and the NDM strategy. To analyze the computational complexity of each component, for a problem f of dimension D, the population size is N. Firstly, the DOW strategy needs to be computed only once in each generation and its computational complexity is O(1). Secondly, the VRL strategy calculates the reward feedback for each individual, its computational complexity is O(N). Then, the computational complexity of the VCS strategy to determine the similarity between learning paradigms is O(N). Finally, the computational complexity of the NDM strategy to generate the neighborhood is O(N^2). Hence, based on the above computational complexity analysis of the main components, in every generation, the computational complexity of the proposed algorithm is O(N^2).

4. Experimental results and discussions

4.1. Experiment setup

In this experiment, 30 functions from CEC2017 [70] are selected to evaluate the performance of the proposed NRLPSO. These functions fall into four categories: Unimodal functions ($f_1 - f_3$), multimodal functions ($f_4 - f_{10}$), hybrid functions ($f_{11} - f_{20}$), and composition functions ($f_{21} - f_{30}$). In addition, 12 functions in CEC2022 [71] are selected to further verify the ability of the algorithm to solve the current complex optimization problems. Among them, f_1 is a unimodal function, ($f_2 - f_5$) are basic functions, ($f_6 - f_8$) are hybrid functions, and ($f_9 - f_{12}$) are composition functions.

In this experiment, the proposed algorithm is compared with certain classical or advanced PSO variants. These variants include SLPSO [72], CLPSO [12], XPSO [73], GLPSO [74], BLPSO [75], BFLPSO [76], HCLPSO [77], HCLDMS-PSO [78], and DSPSO [41]. The details of these algorithms are shown in Table 2. Each function is independently run 51 times according to the CEC standards. The termination condition of all algorithms is the maximum evaluation times, which is set to $D * 10^4$, where D is the dimension.

4.2. Experiment 1: Influences of parameter k

The proposed NRLPSO has the key parameter k (neighborhood size). To determine the optimal value of k in the proposed algorithm, for the 30 functions of CEC2017, multiple experiments are conducted on 30-D, as shown in Table 3.

Table 3 shows that the proposed algorithm is suitable for solving different functions with different neighborhood size. For the unimodal functions ($f_1 - f_3$), when $k = 1$, the algorithm performs well for functions f_1 and f_2. For function f_3, with the exception of $k = 1$, the neighborhood size of other settings can converge to a very good solution. For the simple multimodal functions ($f_4 - f_{10}$), such as function f_4, a smaller neighborhood can obtain a better solution, while a larger neighborhood is not conducive to obtaining a better solution. However, for functions ($f_5 - f_9$), a larger neighborhood can obtain a better solution, while a smaller neighborhood is not conducive to obtaining a better solution. A larger neighborhood is beneficial to the population to obtain more beneficial information, while the smaller neighborhood easily makes the algorithm fall into the local optimum. For the hybrid functions ($f_{11} - f_{19}$), setting different neighborhood size has minimal influence on the accuracy of the solution, indicating that the algorithm is less affected by the neighborhood size in solving the hybridization problem. Similarly, for composition functions ($f_{21} - f_{30}$), the algorithm is less affected by neighborhood size. Note that for function f_{22}, the algorithm performed better than other neighborhood size when $k = 4$. The experimental results show that in the CEC2017 test suites, the neighborhood size $k = 8$ ranks first.

An evaluation of the performance of the NRLPSO algorithm for various neighborhood size, concludes that the influence of neighborhood size k on the performance of NRLPSO depends on the characteristics of the optimization problem. On the unimodal functions, a smaller neighborhood is advantageous to the convergence of the algorithm to a better solution, while for the simple multimodal functions, a larger neighborhood is advantageous to the global search of the algorithm and alleviates the problem of the algorithm premature. However, for hybrid and composition functions, different neighborhood size have minimal effect on the performance of the proposed algorithm.

4.3. Experiment 2: Comparisons of the solution accuracy

The proposed algorithm and comparison algorithms are tested on 30-D CEC2017 test suites and 50-D CEC2017 test suites, respectively. Tables 4 and 5 list the mean value for each function and the best results are denoted in bold. Additionally, the performance of each algorithm is intuitively compared by using the ranking that each algorithm results.

In Table 4, for the unimodal functions ($f_1 - f_3$), the algorithm proposed in this paper ranks third for f_1 and ranks first for f_3, which reflects the effectiveness of the algorithm in solving the unimodal functions. For the multimodal functions ($f_4 - f_{10}$), the proposed algorithm ranks first for function f_4, but the proposed algorithm performs poorly on functions f_6 and f_9. The results of the experiments demonstrate that the NRLPSO algorithm performs well in solving certain multimodal functions. In terms of the hybrid functions ($f_{11} - f_{19}$), the proposed algorithm ranks first for functions ($f_{12} - f_{14}$) and f_{18}, but its performance is poor for functions f_{11} and f_{17}. This finding shows that the NRLPSO algorithm is very effective in solving certain hybrid functions. For the composition functions ($f_{20} - f_{30}$), the NRLPSO algorithm performs well for f_{22}, f_{25}, f_{27}, and f_{30}, reflects the superiority of the NRLPSO algorithm in solving composition functions.

The proposed algorithm is compared with certain classical or advanced PSO variants. The testing outcomes demonstrate that the proposed algorithm overall ranks first in the 30-D CEC2017 functions. It is indicated that the proposed algorithm appears to have outstanding performance.

As shown in Table 5, compared with 30-D, the effect of the proposed algorithm is somewhat reduced on 50-D. For the unimodal functions

Table 2
The information of PSO variants.

Algorithm	Years	Parameter information
CLPSO [12]	2006	$N = 40,\ mega = [0.4, 0.9],\ c_1 = c_2 = 1.49445,\ m = 7,\ p_c = [0.05, 0.5].$
SLPSO [72]	2015	$N = 100,\ \alpha = 0.5,\ \beta = 0.01.$
GLPSO [74]	2015	$N = 50,\ \omega = 0.7298,\ c = 1.49618,\ pm = 0.01,\ sg = 7.$
HCLPSO [77]	2015	$N = 40,\ \omega = [0.2, 0.99],\ c_1 = c_2 = [0.5, 2.5],\ c = [1.5, 3].$
BLPSO [75]	2017	$N = 40,\ \omega = [0.2, 0.9],\ c = 1.496,\ I = 1,\ E = 1.$
DSPSO [57]	2019	$N = 40,\ c_1 = 2.0,\ F_{min} = 0.7.$
XPSO [73]	2020	$N = 50,\ \omega = [0.4, 0.9],\ \eta = 0.2,\ Stag_{max} = 5,\ p = 0.2.$
HCLDMSPSO [78]	2020	$N = 40,\ \omega = [0.29, 0.99],\ w_2 = c_1 = c_2 = [0.5, 2.5],\ p_m = 0.1.$
BFLPSO [76]	2021	$N = 40,\ \omega = [0.2, 0.9],\ c = 1.49445,\ I = E = 1, G = 5.$
NRLPSO	Presented	$N = 40,\ \omega = [0.4, 1.0],\ k = 8.$

($f_1 - f_3$), although the overall ranking does not change much, the convergence accuracy of the algorithm in the function f_2 is decreased. For the multimodal functions ($f_4 - f_{10}$), the overall ranking of the proposed algorithm greatly varies. Especially in function f_4, compared with other comparison algorithms, the proposed algorithm is greatly influenced by dimension. In terms of the hybrid functions ($f_{11} - f_{19}$), surprisingly, the ranking of the proposed algorithm on other hybrid functions rises, with the exception of function f_{18}, which indicates that the NRLPSO algorithm has strong competitiveness in solving high-dimensional hybrid problems. For the composition functions, the proposed algorithm is obviously influenced by dimension. However, note that in functions f_{25}, f_{27} and f_{30}, the proposed algorithm is less affected by the dimension than the other comparison algorithms.

The experimental findings demonstrate that the performance of the proposed algorithm declines on 50-D due to the influence of dimension, in which the unimodal functions are less affected by dimension, and the multimodal functions are more affected by dimension. Note that the NRLPSO algorithm performs better in high dimension for hybrid functions. For certain composition functions, dimension has an impact on the performance of the NRLPSO algorithm as well. But the NRLPSO algorithm has merits in the sensitivity of dimension compared with other comparison algorithms and ranks first in a comprehensive performance.

As shown in Table 6, only the algorithm proposed in this paper can converge to the optimal value in the unimodal function f_1. This finding shows that the algorithm has obvious advantages in solving unimodal functions. For the basic functions ($f_2 - f_5$), the proposed algorithm ranks first in f_3 and ranks in the middle for other multimodal functions. The proposed algorithm is demonstrated to provide merits in solving certain basic functions but no obvious advantages in most of them. Similarly, for the hybrid functions ($f_6 - f_8$), the proposed algorithm achieves an acceptable performance for function f_8, but a mediocre performance for functions f_6 and f_7. For the composition functions ($f_9 - f_{12}$), the proposed algorithm ranks in the top three. It is indicated that the NRLPSO has significant advantages in handling certain challenging composition functions.

This paper uses the latest CEC2022 test suites to verify the ability of the NRLPSO algorithm to solve the current complex optimization problems. The experimental results show that the NRLPSO algorithm overall ranks first in 10-D, which successfully proves that the NRLPSO algorithm achieves excellent performance in solving the latest problems to be optimized.

Table 7 shows that compared with 10-D, the performance of the NRLPSO algorithm is reduced for certain functions. In the unimodal function f_1, although the proposed algorithm ranks first, it cannot converge to the optimal value. In terms of the basic functions ($f_2 - f_5$), the algorithm in this paper is less affected by dimension for functions f_2 and f_4, and its relative ranking rises to first. The proposed algorithm on the f_3 function is greatly influenced by dimension, so it cannot converge to the optimal value, and its ranking drops from first to fifth. For the hybrid functions ($f_6 - f_8$), compared with other comparison algorithms, the proposed algorithm is less affected by the dimension, and its relative ranking on functions f_6 has increased. For composition functions ($f_9 - f_{12}$), the ranking of the proposed algorithm remains unchanged for functions f_9 and f_{12}, but the ranking of the algorithm on functions f_{10} and f_{11} decreases due to the influence of dimension.

The experimental findings indicate that the designed algorithm is sensitive to dimension, and the performance of the proposed algorithm decreases with increasing dimension. But the proposed algorithm ranks first in 20-D, demonstrating that the RNLPSO is more robust than other comparison algorithms.

In addition, the DE mutations are applied to the neighborhood of particles to cause the neighborhood particles to be perturbed, which increases the diversity of neighborhoods and effectively alleviates the problem of the PSO tending to fall into the local optimum. To reflect the difference between PSO and DE, this paper compares the proposed algorithm with LSHADE_SPACMA [79], which was ranked fourth on the CEC2017 single objective optimization track. In terms of performance, for both algorithms, three aspects are analyzed in terms of convergence best, mean, and standard deviation. The use of notation "+" indicates that the algorithm convergence accuracy is better than the other algorithm convergence accuracy by more than one order of magnitude. The use of notation "≈" indicates that the convergence accuracy of the algorithm is in the same order of magnitude as the convergence accuracy of the other algorithm. The use of notation "-" indicates that the algorithm convergence accuracy is inferior to another algorithm by more than one order of magnitude. The experimental results are shown in Table 8.

As can be seen from Table 8, for the best solution, the proposed algorithm in this paper has an advantage only on the function f_4 and function f_{30}, and for the other functions, the two algorithms have similar performance on 13 functions, and LSHADE-SPACMA has an advantage in solving the best solution on 15 functions. In terms of the mean convergence accuracy of the two algorithms, the performance of both is similar on 14 functions, and the performance of the proposed algorithm in this paper needs to be improved on the other 16 problems. In terms of the stability of the algorithm, LSHADE-SPACMA is more stable. In 50-D, the proposed algorithm achieves the best solution on function f_{26} better than LSHADE-SPACMA. The performance of LSHADE-SPACMA decreases on functions f_{16}, f_{17} and f_{26} due to the effect of increasing dimension, and the solution accuracy is similar to that of the proposed algorithm. From the above experimental results, the proposed algorithm is better than most of the PSO variants, but the LSHADE_SPACMA performs better in terms of convergence accuracy.

4.4. Experiment 3: Nonparametric test

To further show the differences between NRLPSO and these PSO variants on the CEC2017 test suits, this paper selects the Wilcoxon signed-rank test [80], a nonparametric statistical analysis method. The purpose of this test is to analyze the differences between the two algorithms.

The p-value provides information about whether a statistical hypothesis test is significant, and also indicates the significance of the result: A smaller p-value indicates a smaller correlation between the compared algorithms. Due to its wide range of employing, statistical

Table 3
Running results at different neighborhood size in the CEC2017 (30-D).

Function		k = 1	k = 2	k = 3	k = 4	k = 5	k = 6	k = 7	k = 8	k = 9	k = 10
f_1	Mean	**6.58E+01**	1.79E+02	1.85E+02	1.48E+02	3.14E+02	1.77E+02	2.74E+02	1.71E+02	1.83E+02	2.47E+02
	Rank	1	5	7	2	10	4	9	3	6	8
f_2	Mean	**9.25E+00**	2.17E+04	1.08E+02	4.37E+04	3.05E+04	2.13E+05	1.76E+05	2.33E+03	2.33E+06	1.22E+06
	Rank	1	4	2	6	5	8	7	3	10	9
f_3	Mean	1.73E−06	1.65E−13	1.57E−13	1.54E−13	**1.49E−13**	**1.49E−13**	**1.49E−13**	**1.49E−13**	1.60E−13	1.64E−13
	Rank	10	9	6	5	1	1	1	1	7	8
f_4	Mean	5.70E+01	**1.55E+01**	2.24E+01	2.46E+01	2.46E+01	2.25E+01	2.70E+01	2.55E+01	2.67E+01	2.71E+01
	Rank	10	1	2	5	4	3	8	6	7	9
f_5	Mean	3.63E+01	2.72E+01	2.59E+01	2.52E+01	2.61E+01	2.35E+01	2.39E+01	2.27E+01	**2.14E+01**	2.28E+01
	Rank	10	9	7	6	8	4	5	2	1	3
f_6	Mean	4.27E−02	8.28E−03	**5.73E−03**	7.75E−03	7.90E−03	1.16E−02	6.21E−03	6.00E−03	1.03E−02	9.15E−03
	Rank	10	6	1	4	5	9	3	2	8	7
f_7	Mean	7.39E+01	6.25E+01	6.00E+01	5.98E+01	5.60E+01	5.31E+01	5.31E+01	5.14E+01	**5.10E+01**	5.23E+01
	Rank	10	9	8	7	6	4	5	2	1	3
f_8	Mean	3.87E+01	3.16E+01	2.79E+01	2.70E+01	2.70E+01	2.56E+01	2.37E+01	**2.34E+01**	2.37E+01	2.38E+01
	Rank	10	9	8	7	6	5	2	1	3	4
f_9	Mean	9.43E+00	5.22E+00	3.46E+00	3.91E+00	2.76E+00	2.63E+00	1.43E+00	**1.37E+00**	1.54E+00	1.46E+00
	Rank	10	9	7	8	6	5	2	1	4	3
f_{10}	Mean	2.23E+03	1.83E+03	1.83E+03	**1.62E+03**	1.69E+03	1.69E+03	1.64E+03	1.63E+03	1.74E+03	1.69E+03
	Rank	10	9	8	1	6	5	3	2	7	4
f_{11}	Mean	7.40E+01	7.91E+01	8.17E+01	7.17E+01	6.88E+01	**6.55E+01**	6.99E+01	6.89E+01	6.94E+01	6.77E+01
	Rank	8	9	10	7	3	1	6	4	5	2
f_{12}	Mean	6.61E+03	**6.44E+03**	9.00E+03	8.19E+03	7.30E+03	9.28E+03	9.66E+03	9.91E+03	1.14E+04	9.80E+03
	Rank	2	1	5	4	3	6	7	9	10	8
f_{13}	Mean	4.18E+02	3.57E+02	**2.67E+02**	4.11E+02	4.37E+02	3.37E+02	5.93E+02	4.13E+02	5.94E+02	5.34E+02
	Rank	6	3	1	4	7	2	9	5	10	8
f_{14}	Mean	**2.55E+02**	3.39E+02	3.68E+02	3.96E+02	4.01E+02	4.54E+02	4.33E+02	4.26E+02	5.84E+02	5.23E+02
	Rank	1	2	3	4	5	8	7	6	10	9
f_{15}	Mean	1.09E+03	7.10E+02	6.00E+02	**4.54E+02**	6.32E+02	5.49E+02	8.33E+02	7.98E+02	6.64E+02	5.75E+02
	Rank	10	7	4	1	5	2	9	8	6	3
f_{16}	Mean	3.18E+02	3.04E+02	2.36E+02	2.65E+02	2.07E+02	2.32E+02	2.60E+02	2.02E+02	2.21E+02	**1.95E+02**
	Rank	10	9	6	8	3	5	7	2	4	1
f_{17}	Mean	1.29E+02	1.33E+02	1.34E+02	1.20E+02	1.00E+02	1.03E+02	8.88E+01	**8.54E+01**	9.45E+01	9.32E+01
	Rank	8	9	10	7	5	6	2	1	4	3
f_{18}	Mean	**1.44E+04**	3.16E+04	3.00E+04	2.96E+04	3.62E+04	4.30E+04	4.37E+04	4.82E+04	5.40E+04	4.14E+04
	Rank	1	4	3	2	5	7	8	9	10	6
f_{19}	Mean	4.93E+02	7.44E+02	**4.32E+02**	6.77E+02	5.58E+02	5.29E+02	6.27E+02	8.94E+02	4.93E+02	8.11E+02
	Rank	2	8	1	7	5	4	6	10	3	9
f_{20}	Mean	9.32E+01	1.16E+02	9.82E+01	8.38E+01	7.78E+01	9.00E+01	7.10E+01	**6.30E+01**	6.49E+01	6.70E+01
	Rank	8	10	9	6	5	7	4	1	2	3
f_{21}	Mean	2.39E+02	2.31E+02	2.28E+02	2.27E+02	2.27E+02	2.25E+02	2.24E+02	2.24E+02	**2.23E+02**	2.24E+02
	Rank	10	9	8	7	6	5	4	2	1	3
f_{22}	Mean	1.00E+02	1.00E+02	1.00E+02	**9.89E+01**	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.00E+02
	Rank	2	2	2	1	2	2	2	2	2	2
f_{23}	Mean	3.85E+02	3.80E+02	3.79E+02	3.76E+02	3.75E+02	**3.70E+02**	3.73E+02	3.73E+02	3.73E+02	3.71E+02
	Rank	10	9	8	7	6	1	5	4	3	2
f_{24}	Mean	4.50E+02	4.45E+02	4.44E+02	4.42E+02	4.42E+02	4.41E+02	4.41E+02	4.41E+02	4.41E+02	**4.39E+02**
	Rank	10	9	8	7	6	3	5	4	2	1
f_{25}	Mean	3.89E+02	**3.77E+02**	3.78E+02	3.78E+02	3.79E+02	3.79E+02	3.79E+02	3.79E+02	3.79E+02	3.80E+02
	Rank	10	1	2	3	4	5	6	8	7	9
f_{26}	Mean	9.10E+02	9.51E+02	1.03E+03	8.47E+02	8.73E+02	1.06E+03	8.97E+02	8.68E+02	8.56E+02	**7.32E+02**
	Rank	7	8	9	2	5	10	6	4	3	1
f_{27}	Mean	5.18E+02	4.98E+02	4.98E+02	5.00E+02	4.98E+02	**4.97E+02**	4.98E+02	4.99E+02	4.99E+02	5.00E+02
	Rank	10	4	3	8	2	1	5	6	7	9
f_{28}	Mean	**3.05E+02**	3.21E+02	3.55E+02	3.44E+02	3.37E+02	3.59E+02	3.45E+02	3.53E+02	3.62E+02	3.34E+02
	Rank	1	2	8	5	4	9	6	7	10	3
f_{29}	Mean	5.61E+02	4.61E+02	4.89E+02	4.64E+02	4.60E+02	4.39E+02	**4.33E+02**	4.63E+02	4.39E+02	4.58E+02
	Rank	10	6	9	8	5	3	1	7	2	4
f_{30}	Mean	6.69E+03	**2.34E+03**	3.07E+03	2.93E+03	2.84E+03	3.12E+03	3.03E+03	2.98E+03	2.75E+03	3.77E+03
	Rank	10	1	7	4	3	8	6	5	2	9
	Total rank	218	183	172	153	146	143	156	**127**	157	153
	Final rank	10	9	8	4	3	2	6	**1**	7	4

software programs can be used to calculate the p-value for the test. IBM SPSS is chosen as the calculation tool. $R+$ be the rank sum of the problem for which the previous algorithm outperforms the back algorithm. $R-$ is the rank sum of problems for which the previous algorithm is weaker than the latter. Grades with $d_i = 0$ are evenly divided between the sums; if they have an odd number, disregard one. The (14) and (15) demonstrate how to calculate the $R+$ and $R-$.

$$R^{+} = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{14}$$

$$R^{-} = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{15}$$

The results of the Wilcoxon nonparametric test for NRLPSO and these compared algorithms are shown in Table 9, where the p-value for NRLPSO is less than 0.05 with most of the comparison algorithms, and only with the BLPSO algorithm is the p-value greater than 0.05. This indicates that NRLPSO outperforms most of the comparison algorithms, but not significantly better than the BLPSO algorithm.

Table 4
Running results of 10 PSO algorithms in the CEC2017 (30-*D*).

Function		SLPSO	CLPSO	XPSO	GLPSO	BLPSO	BFLPSO	HCLPSO	HCLDMS-PSO	DSPSO	NRLPSO
f_1	Mean	4.58E+03	**2.05E+01**	1.90E+06	1.60E+03	2.59E+03	2.04E+03	6.64E+01	1.17E+03	2.07E+03	1.85E+02
	Rank	9	1	10	5	8	6	2	4	7	3
f_2	Mean	**6.18E−05**	1.83E+15	3.60E+19	8.86E−01	1.84E+08	2.19E+08	6.45E−04	3.08E+01	1.53E+09	1.08E+02
	Rank	1	9	10	3	6	7	2	4	8	5
f_3	Mean	6.38E+03	1.38E+04	3.70E+03	6.00E−06	8.97E+02	8.50E+03	1.40E−04	2.58E+01	4.93E−06	**1.57E−13**
	Rank	8	10	7	3	6	9	4	5	2	1
f_4	Mean	7.66E+01	6.93E+01	1.62E+02	5.09E+01	1.04E+02	1.11E+02	5.37E+01	7.94E+01	1.20E+02	**2.24E+01**
	Rank	5	4	10	2	7	8	3	6	9	1
f_5	Mean	2.15E+01	4.94E+01	1.21E+02	4.86E+01	2.21E+01	2.90E+01	4.18E+01	2.81E+01	**1.72E+01**	2.59E+01
	Rank	2	9	10	8	3	6	7	5	1	4
f_6	Mean	4.90E−04	1.61E−11	2.86E+00	2.52E−04	2.28E−07	**1.14E−13**	4.28E−13	1.70E−03	4.77E−05	5.73E−03
	Rank	7	3	10	6	4	1	2	8	5	9
f_7	Mean	1.77E+02	8.58E+01	1.86E+02	7.10E+01	5.69E+01	6.74E+01	7.78E+01	5.64E+01	**4.12E+01**	6.00E+01
	Rank	9	8	10	6	3	5	7	2	1	4
f_8	Mean	1.84E+01	5.79E+01	1.09E+02	4.96E+01	2.36E+01	3.21E+01	4.32E+01	2.79E+01	**1.68E+01**	2.79E+01
	Rank	2	9	10	8	3	6	7	5	1	4
f_9	Mean	1.62E−01	2.01E+01	7.78E+01	2.14E+01	5.68E−02	**1.76E−03**	7.36E+00	8.91E−03	1.69E−01	3.46E+00
	Rank	4	8	10	9	3	1	7	2	5	6
f_{10}	Mean	**1.11E+03**	2.49E+03	4.68E+03	2.52E+03	2.02E+03	2.52E+03	2.27E+03	2.25E+03	1.84E+03	1.83E+03
	Rank	1	7	10	9	4	8	6	5	3	2
f_{11}	Mean	2.98E+01	6.31E+01	1.50E+02	6.73E+01	**2.61E+01**	4.85E+01	5.98E+01	3.29E+01	8.41E+01	8.17E+01
	Rank	2	6	10	7	1	4	5	3	9	8
f_{12}	Mean	5.49E+04	4.73E+05	2.58E+06	2.06E+04	2.38E+05	3.02E+05	2.95E+04	7.13E+04	1.99E+04	**9.00E+03**
	Rank	5	9	10	3	7	8	4	6	2	1
f_{13}	Mean	1.18E+04	3.84E+02	4.24E+04	1.49E+04	1.05E+04	7.89E+03	1.01E+03	3.66E+03	1.00E+04	**2.67E+02**
	Rank	8	2	10	9	7	5	3	4	6	1
f_{14}	Mean	1.80E+04	2.44E+04	1.61E+04	1.54E+03	8.93E+03	5.28E+03	5.81E+03	2.94E+03	6.05E+02	**3.68E+02**
	Rank	9	10	8	3	7	5	6	4	2	1
f_{15}	Mean	3.26E+03	**1.36E+02**	5.81E+03	3.58E+03	1.69E+03	4.80E+02	2.39E+02	2.09E+03	2.81E+03	6.00E+02
	Rank	8	1	10	9	5	3	2	6	7	4
f_{16}	Mean	2.51E+02	5.30E+02	7.06E+02	6.77E+02	**2.05E+02**	2.37E+02	5.53E+02	3.31E+02	5.57E+02	2.36E+02
	Rank	4	6	10	9	1	3	7	5	8	2
f_{17}	Mean	6.73E+01	1.27E+02	1.53E+02	2.23E+02	**4.47E+01**	5.04E+01	1.36E+02	6.83E+01	1.32E+02	1.34E+02
	Rank	3	5	9	10	1	2	8	4	6	7
f_{18}	Mean	1.35E+05	1.47E+05	3.55E+05	4.36E+04	1.04E+05	8.26E+04	8.00E+04	9.83E+04	5.61E+04	**3.00E+04**
	Rank	8	9	10	2	7	5	4	6	3	1
f_{19}	Mean	5.05E+03	**7.88E+01**	6.87E+03	5.47E+03	3.77E+03	1.48E+03	2.08E+02	4.00E+03	4.06E+03	4.32E+02
	Rank	8	1	10	9	5	4	2	6	7	3
f_{20}	Mean	1.22E+02	1.71E+02	2.17E+02	2.42E+02	8.58E+01	**7.05E+01**	1.81E+02	1.56E+02	1.78E+02	9.82E+01
	Rank	4	6	9	10	2	1	8	5	7	3
f_{21}	Mean	2.22E+02	2.49E+02	3.14E+02	2.39E+02	2.23E+02	2.29E+02	2.35E+02	2.28E+02	**2.20E+02**	2.28E+02
	Rank	2	9	10	8	3	6	7	4	1	5
f_{22}	Mean	2.49E+02	2.81E+02	2.98E+02	3.27E+02	2.87E+02	1.00E+02	1.50E+02	1.00E+02	1.00E+02	**1.00E+02**
	Rank	6	7	9	10	8	3	5	2	4	1
f_{23}	Mean	3.71E+02	4.07E+02	4.84E+02	4.04E+02	3.76E+02	3.84E+02	3.97E+02	3.78E+02	**3.70E+02**	3.79E+02
	Rank	2	9	10	8	3	6	7	4	1	5
f_{24}	Mean	4.50E+02	4.71E+02	5.83E+02	4.79E+02	4.46E+02	4.51E+02	4.61E+02	4.46E+02	**4.31E+02**	4.44E+02
	Rank	5	8	10	9	4	6	7	3	1	2
f_{25}	Mean	3.88E+02	3.87E+02	4.14E+02	3.97E+02	3.88E+02	3.87E+02	3.87E+02	3.87E+02	3.92E+02	**3.78E+02**
	Rank	7	3	10	9	6	5	4	2	8	1
f_{26}	Mean	1.21E+03	8.74E+02	1.20E+03	1.00E+03	1.15E+03	1.24E+03	5.53E+02	1.18E+03	**4.92E+02**	1.03E+03
	Rank	9	3	8	4	6	10	2	7	1	5
f_{27}	Mean	5.16E+02	5.13E+02	5.55E+02	5.20E+02	5.05E+02	5.05E+02	5.14E+02	5.13E+02	5.48E+02	**4.98E+02**
	Rank	7	5	10	8	3	2	6	4	9	1
f_{28}	Mean	3.62E+02	4.21E+02	4.59E+02	**3.44E+02**	3.97E+02	4.07E+02	3.48E+02	3.65E+02	3.70E+02	3.55E+02
	Rank	4	9	10	1	7	8	2	5	6	3
f_{29}	Mean	5.11E+02	5.60E+02	7.01E+02	6.11E+02	**4.69E+02**	4.87E+02	5.30E+02	4.87E+02	4.93E+02	4.89E+02
	Rank	6	8	10	9	1	2	7	3	5	4
f_{30}	Mean	4.41E+03	6.28E+03	5.22E+04	4.40E+03	4.67E+03	3.99E+03	3.96E+03	3.73E+03	5.56E+03	**3.07E+03**
	Rank	6	9	10	5	7	4	3	2	8	1
	Total rank	161	193	290	201	138	149	146	131	143	**98**
	Final rank	7	8	10	9	3	6	5	2	4	**1**

4.5. Experiment 4: Comparisons of convergence performance

This experiment is run on the CEC2017 test suites to analyze the convergence performance of the NRLPSO algorithm for four kinds of functions. Fig. 8 shows the convergence curves of the proposed algorithm and nine comparison algorithms for different types of functions. To better highlight the performance of the NRLPSO algorithm, only eight convergence curve figures with representative functions are selected. The convergence curve proves the effectiveness of the strategy proposed in NRLPSO and reflects its good convergence performance. Specific experimental results are shown in the figures below.

Fig. 8 shows the convergence curves of the proposed algorithm and the comparison algorithms for the CEC2017 functions. For the function f_1, the proposed algorithm exhibits a stable convergence trend. For function f_2, the proposed algorithm has no obvious advantage in the early stage of algorithm iteration. In the late stage of algorithm iteration, however, when all comparison algorithms fall into a local optimum, the proposed algorithm can converge to higher precision. For function f_3, the proposed algorithm can quickly converge to a higher precision at the end of algorithm iteration, which indicates that the NRLPSO algorithm has better global search ability compared with all comparison algorithms for unimodal functions. For function f_4,

Table 5
Running results of 10 PSO algorithms in the CEC2017 (50-*D*).

Function		SLPSO	CLPSO	XPSO	GLPSO	BLPSO	BFLPSO	HCLPSO	HCLDMS-PSO	DSPSO	NRLPSO
f_1	Mean	5.60E+03	**6.52E+01**	8.97E+07	2.38E+03	1.87E+03	1.71E+03	2.54E+03	1.23E+03	2.45E+03	1.79E+02
	Rank	9	1	10	6	5	4	8	3	7	2
f_2	Mean	2.08E+06	1.14E+30	2.05E+44	1.35E+12	8.09E+23	1.60E+12	**4.11E+01**	2.13E+13	1.34E+16	1.61E+10
	Rank	2	9	10	4	8	5	1	6	7	3
f_3	Mean	6.29E+04	5.15E+04	1.71E+04	5.95E+03	4.75E+03	3.83E+04	1.34E+01	2.21E+04	1.34E+01	**2.61E−04**
	Rank	10	9	6	5	4	8	2	7	3	1
f_4	Mean	7.27E+01	9.28E+01	3.65E+02	1.06E+02	1.13E+02	1.23E+02	**6.91E+01**	1.14E+02	2.35E+02	8.76E+01
	Rank	2	4	10	5	6	8	1	7	9	3
f_5	Mean	4.99E+01	1.33E+02	2.72E+02	1.13E+02	5.65E+01	6.70E+01	1.02E+02	6.84E+01	**3.37E+01**	5.89E+01
	Rank	2	9	10	8	3	5	7	6	1	4
f_6	Mean	1.97E−03	**3.55E−12**	8.01E+00	2.35E−02	4.85E−06	9.85E−09	1.04E−06	2.84E−02	5.50E−04	4.67E−01
	Rank	6	1	10	7	4	2	3	8	5	9
f_7	Mean	3.83E+02	1.87E+02	3.92E+02	1.49E+02	1.09E+02	1.28E+02	1.61E+02	1.08E+02	**7.26E+01**	1.06E+02
	Rank	9	8	10	6	4	5	7	3	1	2
f_8	Mean	4.29E+01	1.29E+02	2.62E+02	1.10E+02	5.79E+01	7.30E+01	1.03E+02	6.70E+01	**3.43E+01**	6.15E+01
	Rank	2	9	10	8	3	6	7	5	1	4
f_9	Mean	6.11E−01	4.34E+02	6.57E+02	3.99E+02	6.11E−01	6.55E−01	2.37E+02	**3.78E−01**	8.14E−01	2.77E+01
	Rank	2	9	10	8	3	4	7	1	5	6
f_{10}	Mean	**2.08E+03**	4.71E+03	9.01E+03	4.43E+03	4.16E+03	4.64E+03	4.48E+03	4.58E+03	3.03E+03	3.66E+03
	Rank	1	9	10	5	4	8	6	7	2	3
f_{11}	Mean	1.93E+02	8.98E+01	3.88E+02	1.10E+02	**5.70E+01**	6.12E+01	1.32E+02	1.08E+02	1.45E+02	1.63E+02
	Rank	9	3	10	5	1	2	6	4	7	8
f_{12}	Mean	5.18E+05	4.04E+06	1.94E+07	4.30E+05	1.13E+06	1.39E+06	3.58E+05	1.21E+06	3.33E+05	**1.05E+05**
	Rank	5	9	10	4	6	8	3	7	2	1
f_{13}	Mean	3.42E+03	6.11E+02	1.32E+04	2.89E+03	2.10E+03	1.15E+03	2.33E+03	1.97E+03	1.65E+03	**5.89E+02**
	Rank	9	2	10	8	6	3	7	5	4	1
f_{14}	Mean	3.79E+04	3.31E+05	1.35E+05	2.27E+04	8.11E+04	6.76E+04	2.59E+04	3.28E+04	1.16E+04	**6.11E+03**
	Rank	6	10	9	3	8	7	4	5	2	1
f_{15}	Mean	6.09E+03	**2.75E+02**	4.94E+03	6.42E+03	5.53E+03	3.57E+03	9.92E+02	2.01E+03	2.64E+03	6.06E+02
	Rank	9	1	7	10	8	6	3	4	5	2
f_{16}	Mean	5.89E+02	1.10E+03	1.29E+03	1.04E+03	7.63E+02	7.49E+02	1.10E+03	7.54E+02	**5.09E+02**	5.80E+02
	Rank	3	9	10	7	6	4	8	5	1	2
f_{17}	Mean	**3.68E+02**	7.24E+02	1.24E+03	8.33E+02	4.55E+02	4.67E+02	8.03E+02	6.50E+02	5.15E+02	5.97E+02
	Rank	1	7	10	9	2	3	8	6	4	5
f_{18}	Mean	1.78E+05	1.01E+06	2.00E+06	8.93E+04	6.19E+05	3.60E+05	**7.39E+04**	4.66E+05	8.02E+04	1.15E+05
	Rank	5	9	10	3	8	6	1	7	2	4
f_{19}	Mean	1.47E+04	**2.51E+02**	1.48E+04	1.28E+04	1.40E+04	9.51E+03	1.13E+03	8.15E+03	1.43E+04	1.05E+03
	Rank	9	1	10	6	7	5	3	4	8	2
f_{20}	Mean	2.33E+02	5.90E+02	6.79E+02	6.31E+02	2.77E+02	2.97E+02	5.76E+02	4.43E+02	2.19E+02	**1.95E+02**
	Rank	3	8	10	9	4	5	7	6	2	1
f_{21}	Mean	2.94E+02	3.33E+02	5.06E+02	2.98E+02	2.58E+02	2.72E+02	3.03E+02	2.67E+02	**2.36E+02**	2.64E+02
	Rank	6	9	10	7	2	5	8	4	1	3
f_{22}	Mean	2.42E+03	5.02E+03	7.02E+03	4.18E+03	4.38E+03	4.86E+03	4.40E+03	3.25E+03	**4.52E+02**	3.27E+03
	Rank	2	9	10	5	6	8	7	3	1	4
f_{23}	Mean	**4.62E+02**	5.70E+02	7.46E+02	5.41E+02	4.89E+02	5.00E+02	5.47E+02	5.03E+02	4.68E+02	4.92E+02
	Rank	1	9	10	7	3	5	8	6	2	4
f_{24}	Mean	5.59E+02	7.41E+02	8.46E+02	6.13E+02	5.63E+02	5.67E+02	6.30E+02	5.68E+02	**5.26E+02**	5.46E+02
	Rank	3	9	10	7	4	5	8	6	1	2
f_{25}	Mean	5.27E+02	5.43E+02	6.94E+02	5.66E+02	5.29E+02	5.47E+02	5.32E+02	5.18E+02	5.83E+02	**4.54E+02**
	Rank	3	6	10	8	4	7	5	2	9	1
f_{26}	Mean	1.58E+03	2.32E+03	2.04E+03	1.95E+03	1.69E+03	1.71E+03	2.17E+03	1.79E+03	**4.92E+02**	1.65E+03
	Rank	2	10	8	7	4	5	9	6	1	3
f_{27}	Mean	6.49E+02	6.33E+02	7.80E+02	6.99E+02	5.77E+02	5.64E+02	6.34E+02	6.32E+02	7.89E+02	**5.00E+02**
	Rank	7	5	9	8	3	2	6	4	10	1
f_{28}	Mean	4.94E+02	5.25E+02	6.45E+02	5.06E+02	5.00E+02	5.15E+02	4.92E+02	**4.82E+02**	5.37E+02	4.91E+02
	Rank	4	8	10	6	5	7	3	1	9	2
f_{29}	Mean	5.02E+02	7.59E+02	1.22E+03	8.23E+02	**4.47E+02**	4.48E+02	7.25E+02	6.83E+02	6.52E+02	5.94E+02
	Rank	3	8	10	9	1	2	7	6	5	4
f_{30}	Mean	1.03E+06	7.06E+05	4.75E+06	8.24E+05	8.35E+05	7.64E+05	7.76E+05	8.58E+05	4.15E+06	**1.25E+05**
	Rank	8	2	10	5	6	3	4	7	9	1
	Total rank	147	213	242	191	172	145	177	150	126	**87**
	Final rank	4	9	10	8	6	3	7	5	2	**1**

compared with most comparison algorithms, the NRLPSO algorithm has obvious advantages in convergence speed, which can quickly converge to good precision at the early stage of algorithm iteration and can converge to better precision at the late stage of algorithm iteration when all comparison algorithms fall into a local optimum. For functions f_5, f_7, and f_{10}, the proposed algorithm has good exploitation ability, once the algorithm explores good solution, it will develop around the solution and then rapidly converge to better solution. It is shown that the NRLPSO algorithm has better exploration and exploitation ability than all comparison algorithms on multimodal functions. For function f_{13}, the convergence performance shows that the strategy designed in this paper can help the algorithm jump out of the local optimal solution very well. For functions f_{12}, f_{14} and f_{18}, the proposed algorithm converges to better precision in the late stage. As shown in Fig. 8, for functions f_{25} and f_{27}, the proposed algorithm has an obvious advantage in convergence speed, but not in convergence accuracy, and the proposed algorithm will fall into local optimal at the late stage of algorithm iteration. For function f_{30}, the convergence performance reflects the good exploitation ability of the algorithm, and the algorithm uses the optimal solution to exploit a better solution. This finding shows that the performance of the NRLPSO algorithm needs to be further improved for composition functions.

Table 6
Running results of 10 PSO algorithms in the CEC2022 (10-D).

Function		SLPSO	CLPSO	XPSO	GLPSO	BLPSO	BFLPSO	HCLPSO	HCLDMS-PSO	DSPSO	NRLPSO
f_1	Mean	1.11E−15	1.51E+01	3.12E−14	2.00E+02	2.85E−12	1.07E−01	8.21E+03	1.03E−13	6.90E+01	**0.00E+00**
	Rank	2	7	3	9	5	6	10	4	8	1
f_2	Mean	7.86E+00	**1.52E+00**	5.04E+00	2.03E+02	4.05E+00	3.19E+00	1.18E+02	2.63E+00	3.91E+00	3.23E+00
	Rank	8	1	7	10	6	3	9	2	5	4
f_3	Mean	3.20E−07	2.53E−11	2.30E−07	3.00E+02	**0.00E+00**	1.31E−11	2.76E+01	1.48E−08	1.49E+00	**0.00E+00**
	Rank	7	4	6	10	1	3	9	5	8	1
f_4	Mean	6.85E+00	1.01E+01	9.11E+00	4.08E+02	**3.13E+00**	5.37E+00	5.42E+01	4.32E+00	6.82E+00	4.97E+00
	Rank	6	8	7	10	1	4	9	2	5	3
f_5	Mean	1.96E−02	1.32E−01	7.80E−14	4.00E+02	2.75E−12	2.14E−06	3.36E+02	**6.69E−14**	5.72E+00	1.07E−02
	Rank	6	7	2	10	3	4	9	1	8	5
f_6	Mean	1.34E+03	5.76E+01	1.21E+03	2.55E+03	1.64E+02	**4.67E+01**	1.24E+07	2.57E+02	1.07E+05	1.13E+03
	Rank	7	2	6	8	3	1	10	4	9	5
f_7	Mean	9.10E+00	5.47E−02	9.49E+00	1.30E+03	6.32E−01	**1.59E−02**	8.67E+01	1.95E+00	1.31E+01	5.60E+00
	Rank	6	2	7	10	3	1	9	4	8	5
f_8	Mean	1.93E+01	8.15E+00	1.73E+01	1.42E+03	6.65E+00	8.22E+00	4.10E+01	1.31E+01	6.37E+00	**1.03E+00**
	Rank	8	4	7	10	3	5	9	6	2	1
f_9	Mean	2.29E+02	2.29E+02	2.31E+02	1.63E+03	2.29E+02	2.29E+02	3.47E+02	2.29E+02	2.33E+02	**1.86E+02**
	Rank	2	6	7	10	2	5	9	2	8	1
f_{10}	Mean	1.30E+02	**1.06E+01**	1.73E+02	1.51E+03	1.00E+02	1.00E+02	1.19E+02	1.00E+02	1.00E+02	1.00E+02
	Rank	8	1	9	10	5	4	7	2	6	3
f_{11}	Mean	7.37E+01	1.63E+01	8.43E+01	1.50E+03	8.93E+00	1.25E−07	2.25E+02	5.26E−13	2.03E+01	**4.55E−13**
	Rank	7	5	8	10	4	3	9	2	6	1
f_{12}	Mean	1.66E+02	1.65E+02	1.66E+02	1.66E+03	1.64E+02	1.64E+02	1.90E+02	1.65E+02	1.67E+02	**1.47E+02**
	Rank	6	4	4	10	2	2	9	4	8	1
	Total rank	73	51	73	117	38	41	108	38	81	**31**
	Final rank	6	5	6	10	2	4	9	2	8	**1**

Table 7
Running results of 10 PSO algorithms in the CEC2022 (20-D).

Function		SLPSO	CLPSO	XPSO	GLPSO	BLPSO	BFLPSO	HCLPSO	HCLDMS-PSO	DSPSO	NRLPSO
f_1	Mean	2.44E−04	1.04E+03	3.32E−13	2.00E+02	2.10E−01	2.72E+02	3.11E+04	5.26E−06	6.48E+02	**5.46E−14**
	Rank	4	9	2	6	5	7	10	3	8	1
f_2	Mean	5.01E+01	4.34E+01	5.27E+01	2.58E+02	4.99E+01	5.05E+01	3.49E+02	4.73E+01	6.27E+01	**1.52E+01**
	Rank	5	2	7	9	4	6	10	3	8	1
f_3	Mean	1.03E−06	2.50E−11	1.40E−02	3.00E+02	6.25E−09	**1.11E−13**	3.87E+01	4.29E−05	3.24E+00	2.78E−05
	Rank	4	2	7	10	3	1	9	6	8	5
f_4	Mean	5.78E+01	4.87E+01	2.35E+01	4.26E+02	**1.32E+01**	1.71E+01	1.51E+02	1.46E+01	3.51E+01	1.41E+01
	Rank	8	7	5	10	1	4	9	3	6	2
f_5	Mean	5.14E−02	2.69E+01	9.80E−01	4.00E+02	1.76E−03	3.19E−06	1.56E+03	**5.14E−10**	4.27E+01	5.15E−01
	Rank	4	7	6	9	3	2	10	1	8	5
f_6	Mean	2.31E+03	**2.29E+02**	2.62E+03	3.30E+03	1.37E+03	5.24E+02	2.85E+08	1.32E+03	6.01E+06	9.67E+02
	Rank	6	1	7	8	5	2	10	4	9	3
f_7	Mean	2.56E+01	2.58E+01	3.08E+01	1.33E+03	**1.80E+01**	2.20E+01	1.74E+02	2.61E+01	4.09E+01	2.35E+01
	Rank	4	5	7	10	1	2	9	6	8	3
f_8	Mean	2.37E+01	2.23E+01	5.27E+01	1.42E+03	2.18E+01	2.26E+01	8.77E+01	2.17E+01	3.11E+01	**2.05E+01**
	Rank	6	4	8	10	3	5	9	2	7	1
f_9	Mean	1.81E+02	1.81E+02	1.83E+02	1.58E+03	1.81E+02	1.81E+02	3.14E+02	1.81E+02	1.87E+02	**1.65E+02**
	Rank	2	4	7	10	5	6	9	3	8	1
f_{10}	Mean	1.54E+02	**4.49E−01**	1.91E+02	1.51E+03	8.31E+01	8.14E+01	2.05E+02	1.00E+02	1.09E+02	9.88E+01
	Rank	7	1	8	10	3	2	9	5	6	4
f_{11}	Mean	3.14E+02	**2.52E+02**	3.31E+02	1.79E+03	3.08E+02	3.02E+02	1.99E+03	3.00E+02	4.46E+02	2.71E+02
	Rank	6	1	7	9	5	4	10	3	8	2
f_{12}	Mean	2.46E+02	2.44E+02	2.82E+02	1.75E+03	2.38E+02	2.36E+02	4.17E+02	2.47E+02	3.04E+02	**2.00E+02**
	Rank	5	4	7	10	3	2	9	6	8	1
	Total rank	61	47	78	111	41	43	113	45	92	**29**
	Final rank	6	5	7	9	2	3	10	4	8	**1**

4.6. *Experiment 5: Effectiveness of the involved strategies*

In this paper, the effectiveness of the proposed strategies is analyzed on the CEC2017 test suites. Each function is independently run 51 times, and two indicators including the mean and standard deviation are used to analyze the performance of the NRLPSO algorithm. The DOW strategy is not utilized in NRLPSO$_1$, and the inertia weight is set to $\omega = 2$. In NRLPSO$_2$, the proposed VRL strategy is not employed to select the velocity update operator, but the random velocity update operator is selected. NRLPSO$_3$ does not use the proposed VCS strategy to select learning paradigms but uses random learning paradigms. In NRLPSO$_4$, the proposed NDM strategy is not applied to carry out differential mutation in the neighborhood. The experimental results are shown in Table 10.

According to Table 10, in variant NRLPSO$_1$, algorithm performance significantly deteriorates after the DOW strategy is removed, indicating that the proposed DOW strategy has an important role in NRLPSO performance. The convergence accuracy of NRLPSO$_2$ is equivalent to NRLPSO for the function f_{22}, indicating that the VRL strategy proposed on this function has no obvious improvement on NRLPSO performance. The convergence accuracy of NRLPSO$_2$ for function f_{23} is better than NRLPSO, but the improvement is small. In general, the proposed VRL strategy can effectively guide the particle selection of the appropriate speed update operator and improve the performance of NRLPSO. In variant NRLPSO$_3$, with the exception of functions f_6 and f_{27}, the proposed strategy has poor performance, and the convergence accuracy of other functions by using the VCS strategy is improved. In variant NRLPSO$_4$, the algorithm performs poorly for all functions without using

Table 8
Performance comparison of NRLPSO and LSHADE_SPACMA in the CEC2017 (30-*D*).

Function	LSHADE_SPACMA			NRLPSO		
	Best	Mean	Std	Best	Mean	Std
f_1	0.00E+00 (+)	0.00E+00 (+)	0.00E+00 (+)	5.03E−01 (−)	1.85E+02 (−)	1.97E+02 (−)
f_2	0.00E+00 (+)	1.96E−02 (+)	1.40E−01 (+)	5.67E−06 (−)	1.08E+02 (−)	6.58E+03 (−)
f_3	0.00E+00 (+)	0.00E+00 (+)	0.00E+00 (+)	1.14E−13 (−)	1.57E−13 (−)	2.78E−14 (−)
f_4	5.86E+01 (−)	5.86E+01 (≈)	0.00E+00 (+)	4.04E−04 (+)	2.24E+01 (≈)	8.31E+00 (−)
f_5	0.00E+00 (+)	3.45E+00 (+)	2.63E+00 (≈)	1.49E+01 (−)	2.59E+01 (−)	3.90E+00 (≈)
f_6	0.00E+00 (+)	0.00E+00 (+)	0.00E+00 (+)	2.11E−04 (−)	5.73E−03 (−)	1.34E−02 (−)
f_7	3.20E+01 (≈)	3.38E+01 (≈)	8.49E−01 (+)	4.47E+01 (≈)	6.00E+01 (≈)	4.00E+00 (−)
f_8	0.00E+00 (+)	3.20E+00 (+)	1.60E+00 (≈)	1.59E+01 (−)	2.79E+01 (−)	3.28E+00 (≈)
f_9	0.00E+00 (+)	0.00E+00 (+)	0.00E+00 (+)	1.79E−01 (−)	3.46E+00 (−)	9.80E−01 (−)
f_{10}	7.33E+02 (≈)	1.44E+03 (≈)	2.52E+02 (≈)	9.33E+02 (≈)	1.83E+03 (≈)	2.63E+02 (≈)
f_{11}	0.00E+00 (+)	1.78E+01 (≈)	2.48E+01 (≈)	2.00E+01 (−)	8.17E+01 (≈)	1.41E+01 (≈)
f_{12}	1.30E+02 (+)	6.15E+02 (+)	3.13E+02 (+)	2.03E+03 (−)	9.00E+03 (−)	3.93E+03 (−)
f_{13}	0.00E+00 (+)	1.46E+01 (+)	4.99E+00 (+)	5.38E+01 (−)	2.67E+02 (−)	2.46E+02 (−)
f_{14}	2.10E+01 (+)	2.34E+01 (+)	1.69E+00 (+)	1.09E+02 (−)	3.68E+02 (−)	1.82E+02 (−)
f_{15}	1.13E+00 (+)	4.46E+00 (+)	2.29E+00 (+)	1.97E+01 (−)	6.00E+02 (−)	6.03E+02 (−)
f_{16}	2.34E+00 (+)	2.52E+01 (+)	1.75E+01 (≈)	1.32E+01 (−)	2.36E+02 (−)	8.31E+01 (≈)
f_{17}	1.12E+01 (≈)	3.04E+01 (+)	8.12E+00 (+)	4.33E+01 (≈)	1.34E+02 (−)	2.92E+01 (−)
f_{18}	2.04E+01 (+)	2.34E+01 (+)	1.86E+00 (+)	8.23E+03 (−)	3.00E+04 (−)	1.78E+04 (−)
f_{19}	3.15E+00 (+)	1.03E+01 (+)	2.24E+00 (+)	6.10E+01 (−)	4.32E+02 (−)	3.95E+02 (−)
f_{20}	2.47E+01 (≈)	8.38E+01 (≈)	5.55E+01 (≈)	3.42E+01 (≈)	9.82E+01 (≈)	2.22E+01 (≈)
f_{21}	2.00E+02 (≈)	2.07E+02 (≈)	3.52E+00 (≈)	2.16E+02 (≈)	2.28E+02 (≈)	3.31E+00 (≈)
f_{22}	1.00E+02 (≈)	1.00E+02 (≈)	0.00E+00 (+)	1.00E+02 (≈)	1.00E+02 (≈)	6.39E−14 (−)
f_{23}	3.47E+02 (≈)	3.55E+02 (≈)	3.42E+00 (≈)	3.59E+02 (≈)	3.79E+02 (≈)	5.07E+00 (≈)
f_{24}	4.23E+02 (≈)	4.29E+02 (≈)	2.82E+00 (≈)	4.35E+02 (≈)	4.44E+02 (≈)	3.29E+00 (≈)
f_{25}	3.87E+02 (≈)	3.87E+02 (≈)	1.06E−02 (+)	3.77E+02 (≈)	3.78E+02 (≈)	5.66E−01 (−)
f_{26}	8.32E+02 (≈)	9.53E+02 (+)	5.26E+01 (+)	2.00E+02 (≈)	1.03E+03 (−)	4.66E+02 (−)
f_{27}	4.84E+02 (≈)	5.05E+02 (≈)	6.37E+00 (≈)	4.66E+02 (≈)	4.98E+02 (≈)	4.76E+00 (≈)
f_{28}	3.00E+02 (≈)	3.11E+02 (≈)	3.42E+01 (≈)	3.00E+02 (≈)	3.55E+02 (≈)	5.20E+01 (≈)
f_{29}	4.23E+02 (≈)	4.45E+02 (≈)	1.31E+01 (≈)	2.85E+02 (≈)	4.89E+02 (≈)	5.67E+01 (≈)
f_{30}	1.94E+03 (−)	2.01E+03 (≈)	6.28E+01 (+)	4.32E+02 (+)	3.07E+03 (≈)	1.26E+03 (−)
+ \≈ \−	15\13\2	16\14\0	18\12\0	2\13\15	0\14\16	0\12\18

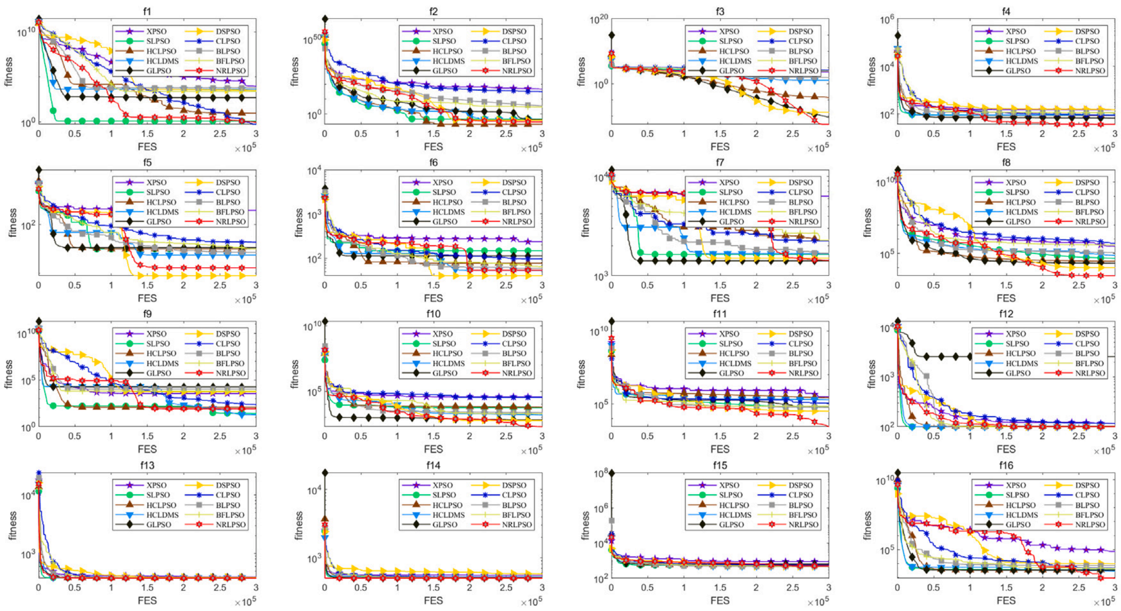


Fig. 8. The convergence curves of NRLPSO with PSO variants in the CEC2017 (30-*D*).

the proposed NDM strategy. The above experimental results show that the proposed strategies have significantly improved the performance of the algorithm. Simultaneously, the stability of the algorithm is greatly improved after the four strategies are employed.

The above experimental analysis proves the effectiveness and correctness of the proposed algorithm. In this paper, the DOW strategy can effectively make particles are able to adapt to their environment and proceed in a more promising direction because of this mode of nonlinear inertia weight. The VRL strategy selects the best velocity update strategy for the present based on the accumulation of historical experience, which helps the particles to make the most consistent movement strategy for itself. The VCS strategy enriches the diversity of the population and effectively alleviates the problem of the algorithm falling into a local optimum too early. In addition, the NDM strategy

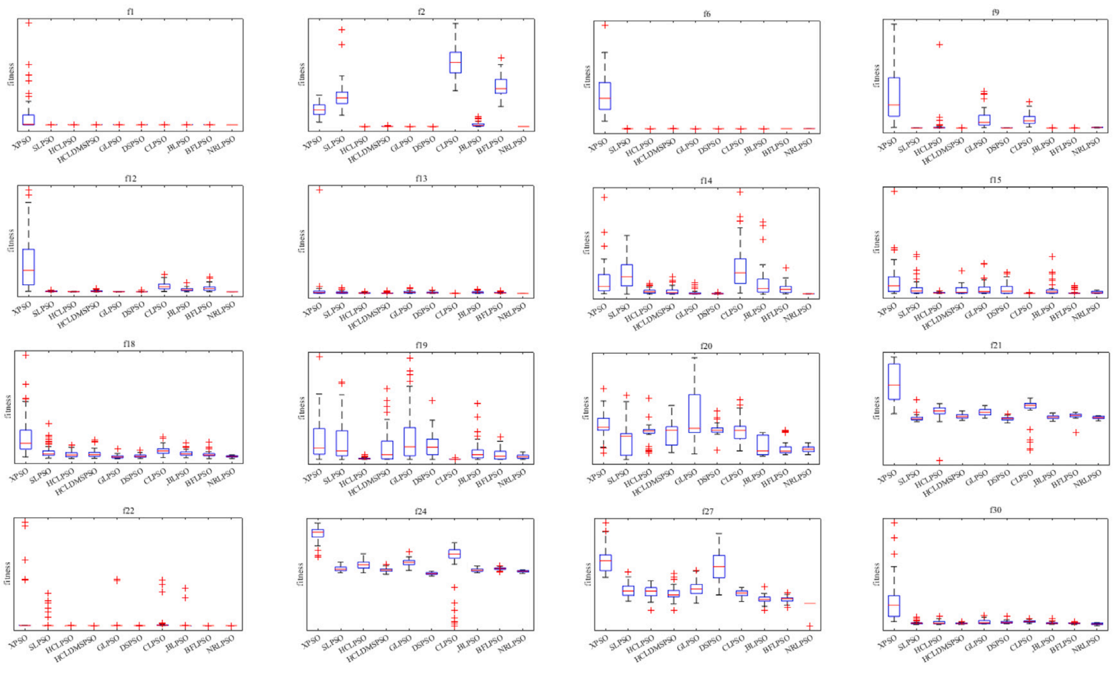


Fig. 9. Stability effect of NRLPSO with PSO variants in the CEC2017 (30-*D*).

Table 9
Nonparametric test results from Table 4.

Comparison	R^+	R^-	p-value
NRLPSO vs. SLPSO	370.5	94.5	4.42E−03
NRLPSO vs. CLPSO	406	59	3.41E−04
NRLPSO vs. XPSO	465	0	2.00E−06
NRLPSO vs. GLPSO	426	39	6.50E−05
NRLPSO vs. BLPSO	326.5	138.5	5.20E−02
NRLPSO vs. BFLPSO	349	116	1.62E−02
NRLPSO vs. HCLPSO	364.5	100.5	6.26E−03
NRLPSO vs. HCLDMS-PSO	348.5	116.5	1.61E−02
NRLPSO vs. DSPSO	343	122	2.23E−02

can fully exploit the neighborhood information so that the particles can effectively jump out of the local optimal.

4.7. Experiment 6: Comparisons of stability performance

The CEC2017 test suites are used to verify the stability of NRLPSO, and representative functions are selected from the four types of functions to illustrate the stability of the NRLPSO algorithm in Fig. 9. The Box-plot is constructed in this paper to analyze the stability of the algorithm. Since each function is independently run for 51 times, the center mark on each box represents the median of the results of 51 runs, and the "+" symbol is used to draw outliers.

For function f_1, with the exception of NRLPSO, the other algorithms have outliers. For function f_2, NRLPSO has high convergence accuracy and the best stability. BFLPSO and NRLPSO perform best on function f_6. For function f_9, SLPSO, DSPSO, and NRLPSO perform well, and there are no outliers. RNLPSO performs best for functions f_{12} and f_{13}, with no outliers and high convergence accuracy. For function f_{14}, NRLPSO has high convergence accuracy and no outliers. For function f_{22}, XPSO, SLPSO, and CLPSO have more outliers, while the other algorithms have better stability. For function f_{27}, NRLPSO has the highest convergence accuracy, but there is an outlier. The above experimental results show that NRLPSO has higher convergence accuracy and better algorithm stability.

5. Conclusion

In this paper, a reinforcement learning-based particle swarm optimization with neighborhood differential mutation strategy is proposed. In the NRLPSO algorithm, the DOW strategy can well balance the global exploration and local exploitation capabilities of the algorithm. The VRL strategy is used to guide the particles to select the appropriate velocity update operators, which improves the optimization ability of the NRLPSO. The VCS strategy can better guide particle movement by selecting appropriate learning paradigms for particles. In addition, the NDM strategy is proposed to enhance the diversity of the neighborhood. The experimental results of NRLPSO and nine PSO variants on CEC2017 and CEC2022 verify that NRLPSO has higher convergence accuracy and better robustness. And the differences between NRLPSO and the comparison algorithms are tested by the Wilcoxon signed-rank test. The statistical analysis results show that the differences between NRLPSO and the compared algorithms are large and the proposed algorithm has significant performance advantages.

This work shows that reinforcement learning has great potential for assisting the PSO to solve optimization problems. In this paper, particles in the population are divided into four states, particles choose the search actions according to their states. The accurate definition of states in the process of evolution could allocate more appropriate operators for particle velocity updates. A discrete action-state space adaptive adjustment velocity update operator is utilized in this paper, whereas a continuous action space adaptive adjustment velocity update operator will be investigated in future work.

CRediT authorship contribution statement

Wei Li: Methodology, Conceptualization. **Peng Liang:** Software, Writing – original draft. **Bo Sun:** Investigation, Software. **Yafeng Sun:** Investigation, Writing – review & editing. **Ying Huang:** Visualization, Supervision.

Table 10
NRLPSO and several variants run results in the CEC2017 (30-*D*).

Function		NRLPSO$_1$	NRLPSO$_2$	NRLPSO$_3$	NRLPSO$_4$	NRLPSO
f_1	Mean	1.95E+04	8.43E+02	1.03E+03	7.04E+02	**1.71E+02**
	Std	8.33E+03	6.44E+02	8.41E+02	5.68E+02	1.53E+02
f_2	Mean	1.15E+27	8.04E+07	3.92E+09	4.49E+15	**2.33E+03**
	Std	1.32E+27	2.68E+08	1.11E+10	9.78E+15	7.50E+03
f_3	Mean	3.88E+04	1.81E−13	7.78E−08	2.13E+01	**1.49E−13**
	Std	6.29E+03	3.54E−14	6.99E−08	1.13E+01	2.78E−14
f_4	Mean	1.66E+02	3.09E+01	3.57E+01	1.58E+02	**2.55E+01**
	Std	2.83E+01	9.83E+00	1.05E+01	1.28E+01	1.06E+01
f_5	Mean	2.35E+02	2.86E+01	2.35E+01	7.17E+01	**2.27E+01**
	Std	1.06E+01	4.85E+00	4.03E+00	1.36E+01	3.34E+00
f_6	Mean	9.65E+00	2.18E−02	**4.41E−03**	1.13E+01	6.00E−03
	Std	2.73E+00	2.10E−02	5.72E−03	2.18E+00	4.97E−03
f_7	Mean	1.54E+02	5.53E+01	5.17E+01	1.33E+02	**5.14E+01**
	Std	2.22E+01	3.59E+00	3.64E+00	1.81E+01	2.96E+00
f_8	Mean	2.32E+02	2.71E+01	2.47E+01	7.87E+01	**2.34E+01**
	Std	1.03E+01	4.65E+00	4.59E+00	1.21E+01	2.93E+00
f_9	Mean	9.39E+02	2.41E+00	2.12E+00	4.23E+02	**1.37E+00**
	Std	4.53E+02	1.57E+00	1.22E+00	1.74E+02	6.86E−01
f_{10}	Mean	6.53E+03	2.02E+03	2.93E+03	3.67E+03	**1.63E+03**
	Std	2.47E+02	3.64E+02	1.66E+03	7.90E+02	3.45E+02
f_{11}	Mean	7.17E+02	8.37E+01	8.55E+01	2.14E+02	**6.89E+01**
	Std	1.01E+02	1.73E+01	1.72E+01	4.92E+01	1.51E+01
f_{12}	Mean	1.59E+08	1.91E+04	2.87E+04	9.61E+06	**9.91E+03**
	Std	5.31E+07	1.04E+04	2.11E+04	8.80E+06	4.60E+03
f_{13}	Mean	3.28E+07	1.28E+03	1.86E+03	7.75E+04	**4.13E+02**
	Std	7.44E+06	1.15E+03	1.93E+03	3.42E+04	2.94E+02
f_{14}	Mean	4.50E+04	8.60E+02	1.99E+03	8.26E+03	**4.26E+02**
	Std	1.95E+04	5.04E+02	8.81E+02	5.27E+03	1.50E+02
f_{15}	Mean	4.89E+06	3.42E+03	2.58E+03	4.51E+04	**7.98E+02**
	Std	1.66E+06	2.94E+03	2.11E+03	1.69E+04	7.07E+02
f_{16}	Mean	1.47E+03	3.06E+02	2.89E+02	7.07E+02	**2.02E+02**
	Std	1.54E+02	9.59E+01	1.01E+02	2.14E+02	8.81E+01
f_{17}	Mean	5.81E+02	1.48E+02	1.27E+02	2.41E+02	**8.54E+01**
	Std	8.54E+01	5.32E+01	4.59E+01	6.04E+01	1.93E+01
f_{18}	Mean	8.05E+05	8.45E+04	1.29E+05	2.03E+05	**4.82E+04**
	Std	2.79E+05	3.53E+04	5.07E+04	7.70E+04	1.68E+04
f_{19}	Mean	1.11E+07	3.23E+03	2.08E+03	7.98E+04	**8.94E+02**
	Std	3.44E+06	2.72E+03	1.58E+03	3.84E+04	6.11E+02
f_{20}	Mean	4.60E+02	1.31E+02	1.13E+02	2.67E+02	**6.30E+01**
	Std	6.96E+01	5.24E+01	4.83E+01	7.82E+01	1.68E+01
f_{21}	Mean	4.26E+02	2.27E+02	2.25E+02	2.75E+02	**2.24E+02**
	Std	1.13E+01	4.30E+00	6.42E+00	1.12E+01	3.95E+00
f_{22}	Mean	3.18E+03	**1.00E+02**	**1.00E+02**	3.83E+02	**1.00E+02**
	Std	3.13E+03	9.50E−01	9.52E−01	8.67E+02	8.92E−14
f_{23}	Mean	5.71E+02	**3.72E+02**	3.73E+02	4.35E+02	3.73E+02
	Std	1.12E+01	3.65E+01	6.28E+00	1.79E+01	4.52E+00
f_{24}	Mean	6.30E+02	4.46E+02	4.43E+02	5.11E+02	**4.41E+02**
	Std	8.48E+00	5.63E+00	3.74E+00	2.60E+01	3.12E+00
f_{25}	Mean	4.21E+02	3.80E+02	3.80E+02	4.12E+02	**3.79E+02**
	Std	1.15E+01	9.29E−01	8.99E−01	1.31E+01	5.91E−01
f_{26}	Mean	3.55E+03	1.02E+03	9.82E+02	1.86E+03	**8.68E+02**
	Std	1.24E+02	4.53E+02	4.12E+02	4.93E+02	4.60E+02
f_{27}	Mean	5.00E+02	5.00E+02	**4.98E+02**	5.56E+02	4.99E+02
	Std	1.30E−04	1.76E+00	9.97E+00	1.08E+01	3.92E+00
f_{28}	Mean	5.00E+02	3.89E+02	3.83E+02	4.59E+02	**3.53E+02**
	Std	2.58E+00	5.06E+01	6.05E+01	3.12E+01	5.22E+01
f_{29}	Mean	1.18E+03	5.02E+02	4.67E+02	8.67E+02	**4.63E+02**
	Std	1.26E+02	6.10E+01	6.20E+01	1.23E+02	3.71E+01
f_{30}	Mean	3.50E+06	4.76E+03	7.34E+03	1.02E+06	**2.98E+03**
	Std	1.24E+06	1.94E+03	3.11E+03	8.40E+05	1.13E+03

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.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant Nos. 62066019, 61903089), the Natural Science Foundation of Jiangxi Province, China (Grant Nos. 20202BABL202020, 20202BAB202014).

References

[1] Ying Lin, Ye-Shi Jiang, Yue-Jiao Gong, Zhi-Hui Zhan, Jun Zhang, A discrete multiobjective particle swarm optimizer for automated assembly of parallel cognitive diagnosis tests, IEEE Trans. Cybern. 49 (7) (2018) 2792–2805.

[2] Weibo Liu, Zidong Wang, Yuan Yuan, Nianyin Zeng, Kate Hone, Xiaohui Liu, A novel sigmoid-function-based adaptive weighted particle swarm optimizer, IEEE Trans. Cybern. 51 (2) (2019) 1085–1093.
[3] Yang Liu, Qinglin Cheng, Yifan Gan, Yuxin Wang, Zhidong Li, Jian Zhao, Multi-objective optimization of energy consumption in crude oil pipeline transportation system operation based on exergy loss analysis, Neurocomputing 332 (2019) 100–110.
[4] Baoye Song, Zidong Wang, Lei Zou, On global smooth path planning for mobile robots using a novel multimodal delayed PSO algorithm, Cogn. Comput. 9 (1) (2017) 5–17.
[5] Zi-Jia Wang, Zhi-Hui Zhan, Ying Lin, Wei-Jie Yu, Hua Wang, Sam Kwong, Jun Zhang, Automatic niching differential evolution with contour prediction approach for multimodal optimization problems, IEEE Trans. Evol. Comput. 24 (1) (2019) 114–128.
[6] Sun-Chong Wang, Interdisciplinary Computing in Java Programming, Vol. 743, Springer Science & Business Media, 2003.
[7] Dervis Karaboga, Artificial bee colony algorithm, Scholarpedia 5 (3) (2010) 6915.
[8] Kenneth V. Price, Differential evolution: a fast and simple numerical optimizer, in: Proceedings of North American Fuzzy Information Processing, IEEE, 1996, pp. 524–527.
[9] Nicholas Metropolis, A.W.R.M.N. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Simulated annealing, J. Chem. Phys. 21 (161–162) (1953) 1087–1092.
[10] Nada M.A. Al Salami, Ant colony optimization algorithm, UbiCC J. 4 (3) (2009) 823–826.
[11] James Kennedy, Russell Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks, Vol. 4, IEEE, 1995, pp. 1942–1948.
[12] Jing J. Liang, A. Kai Qin, Ponnuthurai N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (3) (2006) 281–295.
[13] Rui Mendes, James Kennedy, José Neves, The fully informed particle swarm: simpler, maybe better, IEEE Trans. Evol. Comput. 8 (3) (2004) 204–210.
[14] Bo-Yang Qu, Ponnuthurai Nagaratnam Suganthan, Swagatam Das, A distance-based locally informed particle swarm model for multimodal optimization, IEEE Trans. Evol. Comput. 17 (3) (2012) 387–402.
[15] Maurice Clerc, James Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (1) (2002) 58–73.
[16] Asanga Ratnaweera, Saman K. Halgamuge, Harry C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Trans. Evol. Comput. 8 (3) (2004) 240–255.
[17] Bo-Yang Qu, Ponnuthurai Nagaratnam Suganthan, Jane-Jing Liang, Differential evolution with neighborhood mutation for multimodal optimization, IEEE Trans. Evol. Comput. 16 (5) (2012) 601–614.
[18] Yuhui Shi, Russell C. Eberhart, Empirical study of particle swarm optimization, in: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Vol. 3, IEEE, 1999, pp. 1945–1950.
[19] Feng Wang, Heng Zhang, Aimin Zhou, A particle swarm optimization algorithm for mixed-variable optimization problems, Swarm Evol. Comput. 60 (2021) 100808.
[20] Chia-Feng Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, IEEE Trans. Syst. Man Cybern. B 34 (2) (2004) 997–1006.
[21] P.S. Shelokar, Patrick Siarry, Valadi K. Jayaraman, Bhaskar D. Kulkarni, Particle swarm and ant colony algorithms hybridized for improved continuous optimization, Appl. Math. Comput. 188 (1) (2007) 129–142.
[22] Nianyin Zeng, Y.S. Hung, Yurong Li, Min Du, A novel switching local evolutionary PSO for quantitative analysis of lateral flow immunoassay, Expert Syst. Appl. 41 (4) (2014) 1708–1715.
[23] Nianyin Zeng, Hong Zhang, Yanping Chen, Binqiang Chen, Yurong Liu, Path planning for intelligent robot based on switching local evolutionary PSO algorithm, Assem. Autom. (2016).
[24] Frans Van den Bergh, Andries P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 225–239.
[25] James Kennedy, Rui Mendes, Population structure and particle swarm performance, in: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600), Vol. 2, IEEE, 2002, pp. 1671–1676.
[26] Jane-Jing Liang, Ponnuthurai Nagaratnam Suganthan, Dynamic multi-swarm particle swarm optimizer, in: Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005, IEEE, 2005, pp. 124–129.
[27] Bo-Yang Qu, Ponnuthurai Nagaratnam Suganthan, Swagatam Das, A distance-based locally informed particle swarm model for multimodal optimization, IEEE Trans. Evol. Comput. 17 (3) (2012) 387–402.
[28] Wei Li, Bo Sun, Ying Huang, Soroosh Mahmoodi, Adaptive complex network topology with fitness distance correlation framework for particle swarm optimization, Int. J. Intell. Syst. 37 (8) (2022) 5217–5247.
[29] Nianyin Zeng, Zidong Wang, Weibo Liu, Hong Zhang, Kate Hone, Xiaohui Liu, A dynamic neighborhood-based switching particle swarm optimization algorithm, IEEE Trans. Cybern. (2020).
[30] Xiaodong Li, Xin Yao, Cooperatively coevolving particle swarms for large scale optimization, IEEE Trans. Evol. Comput. 16 (2) (2011) 210–224.
[31] Feng Wang, Xujie Wang, Shilei Sun, A reinforcement learning level-based particle swarm optimization algorithm for large-scale optimization, Inf. Sci. 602 (2022) 298–312.
[32] Gary G. Yen, Moayed Daneshyari, Diversity-based information exchange among multiple swarms in particle swarm optimization, Int. J. Comput. Intell. Appl. 7 (01) (2008) 57–75.
[33] Weilin Du, Bin Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, Inform. Sci. 178 (15) (2008) 3096–3109.
[34] Ran Cheng, Chaoli Sun, Yaochu Jin, A multi-swarm evolutionary framework based on a feedback mechanism, in: 2013 IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 718–724.
[35] Wei Li, Xiang Meng, Ying Huang, Zhang-Hua Fu, Multipopulation cooperative particle swarm optimization with a mixed mutation strategy, Inform. Sci. 529 (2020) 179–196.
[36] Leilei Cao, Lihong Xu, Erik D. Goodman, A neighbor-based learning particle swarm optimizer with short-term and long-term memory for dynamic optimization problems, Inform. Sci. 453 (2018) 463–485.
[37] Hao-Ran Liu, Jing-Chuang Cui, Ze-Dan Lu, Da-Yan Liu, Yu-Jing Deng, A hierarchical simple particle swarm optimization with mean dimensional information, Appl. Soft Comput. 76 (2019) 712–725.
[38] Ying Huang, et al., A fitness landscape ruggedness multiobjective differential evolution algorithm with a reinforcement learning strategy, Appl. Soft Comput. 96 (2020) 106693.
[39] Qiang Yang, Wei-Neng Chen, Tianlong Gu, Huaxiang Zhang, Jeremiah D Deng, Yun Li, Jun Zhang, Segment-based predominant learning swarm optimizer for large-scale optimization, IEEE Trans. Cybern. 47 (9) (2016) 2896–2910.
[40] Tim Blackwell, James Kennedy, Impact of communication topology in particle swarm optimization, IEEE Trans. Evol. Comput. 23 (4) (2018) 689–702.
[41] Xinming Zhang, Xia Wang, Qiang Kang, Jinfeng Cheng, Differential mutation and novel social learning particle swarm optimization algorithm, Inform. Sci. 480 (2019) 109–129.
[42] Afnizanfaizal Abdullah, Safaai Deris, Mohd Saberi Mohamad, Siti Zaiton Mohd Hashim, A new particle swarm evolutionary optimization for parameter estimation of biological models, Int. J. Comput. Inf. Syst. Ind. Manage. Appl. 5 (2013) 571–580.
[43] Afnizanfaizal Abdullah, Safaai Deris, Siti Zaiton Mohd Hashim, Mohd Saberi Mohamad, Satya Nanda Vel Arjunan, An improved local best searching in particle swarm optimization using differential evolution, in: 2011 11th International Conference on Hybrid Intelligent Systems, HIS, IEEE, 2011, pp. 115–120.
[44] Richard S. Sutton, Andrew G. Barto, Reinforcement Learning: an Introduction, Vol. 22447, MIT Press, Cambridge, MA, 1998.
[45] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller, Playing atari with deep reinforcement learning, 2013, arXiv preprint arXiv:1312.5602.
[46] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[47] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al., Mastering the game of go with deep neural networks and tree search, Nature 529 (7587) (2016) 484–489.
[48] Peter Dayan, C.J.C.H. Watkins, Q-learning, Mach. Learn. 8 (3) (1992) 279–292.
[49] Christopher John Cornish Hellaby Watkins, Learning from delayed rewards, 1989.
[50] Gavin A. Rummery, Mahesan Niranjan, On-Line Q-Learning using Connectionist Systems, Vol. 37, Citeseer, 1994.
[51] Richard S. Sutton, Learning to predict by the methods of temporal differences, Mach. Learn. 3 (1) (1988) 9–44.
[52] Satinder P. Singh, Richard S. Sutton, Reinforcement learning with replacing eligibility traces, Mach. Learn. 22 (1) (1996) 123–158.
[53] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, Martin Riedmiller, Deterministic policy gradient algorithms, in: International Conference on Machine Learning, PMLR, 2014, pp. 387–395.
[54] Pratyusha Rakshit, Amit Konar, Pavel Bhowmik, Indrani Goswami, Swagatam Das, Lakhmi C. Jain, Atulya K Nagar, Realization of an adaptive memetic algorithm using differential evolution and Q-learning: A case study in multirobot path planning, IEEE Trans. Syst. Man Cybern. 43 (4) (2013) 814–831.
[55] Grigoris S. Piperagkas, George Georgoulas, Konstantinos E. Parsopoulos, Chrysostomos D. Stylios, Aristidis C. Likas, Integrating particle swarm optimization with reinforcement learning in noisy problems, in: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation, 2012, pp. 65–72.
[56] Licheng Jiao, Maoguo Gong, Shuang Wang, Biao Hou, Zhi Zheng, Qiaodi Wu, Natural and remote sensing image segmentation using memetic computing, IEEE Comput. Intell. Mag. 5 (2) (2010) 78–91.
[57] Hussein Samma, Chee Peng Lim, Junita Mohamad Saleh, A new reinforcement learning-based memetic particle swarm optimizer, Appl. Soft Comput. 43 (2016) 276–297.
[58] Weikang Ning, Baolong Guo, Xinxing Guo, Cheng Li, Yunyi Yan, Reinforcement learning aided parameter control in multi-objective evolutionary algorithm based on decomposition, Prog. Artif. Intell. 7 (4) (2018) 385–398.

[59] Yue Xu, Dechang Pi, A reinforcement learning-based communication topology in particle swarm optimization, Neural Comput. Appl. 32 (14) (2020) 10007–10032.
[60] Ye Tian, Xiaopeng Li, Haiping Ma, Xingyi Zhang, Kay Chen Tan, Yaochu Jin, Deep reinforcement learning based adaptive operator selection for evolutionary multi-objective optimization, IEEE Trans. Emerg. Topics Comput. Intell. (2022).
[61] Byung Il Kwak, Mee Lan Han, Huy Kang Kim, Cosine similarity based anomaly detection methodology for the CAN bus, Expert Syst. Appl. 166 (2021) 114066.
[62] Rabindra Mohanty, Subham Sahoo, Ashok Kumar Pradhan, Frede Blaabjerg, A cosine similarity-based centralized protection scheme for dc microgrids, IEEE J. Emerg. Sel. Top. Power Electron. 9 (5) (2021) 5646–5656.
[63] Tahir Mahmood, Muhammad Ilyas, Zeeshan Ali, Abdu Gumaei, Spherical fuzzy sets-based cosine similarity and information measures for pattern recognition and medical diagnosis, IEEE Access 9 (2021) 25835–25842.
[64] Frans Van Den Bergh, et al., An Analysis of Particle Swarm Optimizers (Ph.D. thesis), University of Pretoria, 2007.
[65] Russ C. Eberhart, Yuhui Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512), Vol. 1, IEEE, 2000, pp. 84–88.
[66] Hao Liu, Xu-Wei Zhang, Liang-Ping Tu, A modified particle swarm optimization using adaptive strategy, Expert Syst. Appl. 152 (2020) 113353.
[67] Amitava Chatterjee, Patrick Siarry, Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization, Comput. Oper. Res. 33 (3) (2006) 859–871.
[68] P.K. Das, H.S. Behera, B.K. Panigrahi, Intelligent-based multi-robot path planning inspired by improved classical Q-learning and improved particle swarm optimization with perturbed velocity, Eng. Sci. Technol. Int. J. 19 (1) (2016) 651–669.
[69] Gang Qian, Shamik Sural, Yuelong Gu, Sakti Pramanik, Similarity between Euclidean and cosine angle distance for nearest neighbor queries, in: Proceedings of the 2004 ACM Symposium on Applied Computing, 2004, pp. 1232–1237.
[70] N.H. Awad, M. Ali, J. Liang, B. Qu, P. Suganthan, Problem Definitions, Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, Tech. Rep., 2016.
[71] Danial Yazdani, Juergen Branke, Mohammad Nabi Omidvar, Xiaodong Li, Changhe Li, Michalis Mavrovouniotis, Trung Thanh Nguyen, Shengxiang Yang, Xin Yao, IEEE CEC 2022 competition on dynamic optimization problems generated by generalized moving peaks benchmark, 2021, arXiv:2106.06174.
[72] Ran Cheng, Yaochu Jin, A social learning particle swarm optimization algorithm for scalable optimization, Inform. Sci. 291 (2015) 43–60.
[73] Xuewen Xia, Ling Gui, Guoliang He, Bo Wei, Yinglong Zhang, Fei Yu, Hongrun Wu, Zhi-Hui Zhan, An expanded particle swarm optimization based on multi-exemplar and forgetting ability, Inform. Sci. 508 (2020) 105–120.
[74] Yue-Jiao Gong, Jing-Jing Li, Yicong Zhou, Yun Li, Henry Shu-Hung Chung, Yu-Hui Shi, Jun Zhang, Genetic learning particle swarm optimization, IEEE Trans. Cybern. 46 (10) (2015) 2277–2290.
[75] Xu Chen, Huaglory Tianfield, Congli Mei, Wenli Du, Guohai Liu, Biogeography-based learning particle swarm optimization, Soft Comput. 21 (24) (2017) 7519–7541.
[76] Xu Chen, Hugo Tianfield, Wenli Du, Bee-foraging learning particle swarm optimization, Appl. Soft Comput. 102 (2021) 107134.
[77] Nandar Lynn, Ponnuthurai Nagaratnam Suganthan, Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation, Swarm Evol. Comput. 24 (2015) 11–24.
[78] Shengliang Wang, Genyou Liu, Ming Gao, Shilong Cao, Aizhi Guo, Jiachen Wang, Heterogeneous comprehensive learning and dynamic multi-swarm particle swarm optimizer with two mutation operators, Inform. Sci. 540 (2020) 175–201.
[79] Ali W. Mohamed, Anas A. Hadi, Anas M. Fattouh, Kamal M. Jambi, LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems, in: 2017 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2017, pp. 145–152.
[80] Joaquín Derrac, Salvador García, Daniel Molina, Francisco Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. 1 (1) (2011) 3–18.