A Dynamic-Neighborhood-Based Switching Particle Swarm Optimization Algorithm

Nianyin Zeng, Zidong Wang, *Fellow*, *IEEE*, Weibo Liu, Hong Zhang, Kate Hone and Xiaohui Liu

***Abstract*—In this paper, a dynamic-neighborhood-based switching PSO (DNSPSO) algorithm is proposed where a new velocity updating mechanism is designed to adjust the personal best position and the global best position according to a distance-based dynamic neighborhood to make full use of the population evolution information among the entire swarm. In addition, a novel switching learning strategy is introduced to adaptively select the acceleration coefficients and update the velocity model according to the searching state at each iteration, thereby contributing to a thorough search of the problem space. Furthermore, the differential evolution algorithm is successfully hybridized with the PSO algorithm to alleviate premature convergence. A series of commonly used benchmark functions (including unimodal, multimodal and rotated multimodal cases) are utilized to comprehensively evaluate the performance of the DNSPSO algorithm. Experimental results demonstrate that the developed DNSPSO algorithm outperforms a number of existing PSO algorithms in terms of the solution accuracy and convergence performance, especially for complicated multimodal optimization problems.**

***Index Terms*—Particle swarm optimization, switching strategy, dynamic neighborhood, topology, differential evolution.**

I. Introduction

The past few decades have witnessed the rapid development of optimization techniques owing to their clear application potential in various research fields including healthcare, engineering, and telecommunications [19], [22], [24], [36], [40]. As a powerful family of optimization techniques, evolutionary computation (EC) approaches have shown outstanding performance in solving optimization problems. Some popular EC approaches include genetic algorithms, Tabu search, simulated annealing, particle swarm optimization (PSO), etc [19], [21], [24], [25], [44]. Compared with other EC algorithms, the PSO algorithm proposed in [15] demonstrates competitive or even superior performance due to its easy implementation, fast convergence rate, guaranteed diversity and satisfactory accuracy [11], [23], [39], [42].

This work was supported in part by the Natural Science Foundation of China under Grants 61873148, 61933007 and 62073271, in part by the Korea Foundation for Advanced Studies, in part by International Science and Technology Cooperation Project of Fujian Province of China under Grant 2019I0003, in part by the Fundamental Research Funds for the Central Universities of China under Grant 20720190009, in part by the Open Fund of Engineering Research Center of Big Data Application in Private Health Medicine of China under Grant KF2020002, and in part by The Open Fund of Provincial Key Laboratory of Eco-Industrial Green Technology-Wuyi University of China. (*Corresponding author: Zidong Wang*)

N. Zeng and H. Zhang are with the Department of Instrumental and Electrical Engineering, Xiamen University, Xiamen, Fujian 361005, P.R. China. Email: zny@xmu.edu.cn

Z. Wang, W. Liu, K. Hone and X. Liu are with the Department of Computer Science, Brunel University London, Uxbridge, Middlesex, UB8 3PH, United Kingdom. Email: Zidong.Wang@brunel.ac.uk

Inspired by the mimicry of social behavior (e.g., birds flocking and fish schooling), the PSO algorithm explores the problem space by updating the particles' velocities and positions based on the swarm intelligence. In fact, the PSO algorithm has become an attractive optimization technique owing to its rather powerful ability to effectively searching the global optimal solution. In this regard, the PSO algorithm has been widely studied and successfully applied to various research fields [4], [13], [27], [48], [49]. Nevertheless, it has been found that the PSO algorithm suffers from certain shortcomings of premature convergence and poor convergence performance when there exist a large number of local optima, especially in high-dimensional optimization problems [18], [26], [29]. As such, great efforts have been made in recent years to develop variant PSO algorithms which can be categorized into four types: 1) modifying parameters [6], [30], [33], [34], [38]; 2) hybridizing with other EC algorithms [14], [32], [42], [45]; 3) using topology for velocity updating [2], [16]–[18], [26], [29]; 4) introducing control theories for designing new learning strategies [10], [20], [28], [35], [47].

Recently, developing PSO algorithms with different topologies and learning strategies has become a research hotspot [3]. Generally, these types of PSO variants aim to enhance the population diversity of the PSO algorithm and alleviate the premature convergence problem. In particular, five well-known topological structures (including all, ring, clusters, pyramid, and von Neumann) have been discussed in [16]. A fully informed PSO algorithm has been introduced in [26] where the velocity of each particle is updated depending on its neighbors' information. In [18], a comprehensive learning PSO algorithm has been developed where each dimension of a single particle is updated by learning from the local best position of different particles. In addition, a locally informed PSO algorithm has been proposed in [29] where the particles are guided to explore the problem space by several personal best positions chosen in the neighborhood.

The developed PSO variants that employ the above mentioned topologies have demonstrated satisfactory performance in searching the global optimal solution. Nevertheless, the topology of these PSO variants is determined at the initialization stage and remains unchanged during the evolution process, which indicates that the neighborhood of the particles is fixed. In this case, the PSO algorithms with a fixed neighborhood have the following two limitations: 1) it is difficult to seek an ideal topology for most optimization problems; and 2) the fixed neighborhood may lead to poor local search performance.

To solve the above mentioned problems, a distance-based

dynamic neighborhood is proposed in this paper to update the personal best position $pbest$ and the global best position $gbest$ so as to strengthen the capability of escaping from the local optima. The conventional PSO algorithm updates the velocity and position of each particle only based on the information of the $pbest$ and the $gbest$ discovered by the entire swarm. That is, the searching strategy of the conventional PSO algorithm restricts the social learning part only to the $gbest$ which may cause the premature convergence problem. In addition, there is no guarantee that the $gbest$ is always close to the global optimum, especially for the complex optimization problems with a large number of local optimal solutions. To make full use of the neighborhood information of other particles, a new updating mechanism of $pbest$ and $gbest$ is proposed in this paper based on the developed dynamic neighborhood. Notice that the evolutionary process of the swarm can be divided into four searching states, including convergence, exploration, exploitation, and jumping out [46]. To further enhance the search ability of the PSO algorithm, a novel switching learning strategy is introduced to adaptively select the acceleration coefficients and adjust the velocity updating model according to the searching state determined by an evolutionary factor at each iteration. In addition, the differential evolution (DE) algorithm [45] is utilized to further enhance the diversities of $pbest$ and $gbest$.

Motivated by above discussions, our aim is to put forward a dynamic-neighborhood-based switching PSO (DNSPSO) algorithm with the aim of improving the search performance of the PSO algorithm. The key contributions of this study can be summarized as follows: 1) a DNSPSO algorithm is proposed where a distance-based dynamic neighborhood is put forward to make full use of the neighborhood information by updating the $pbest$ and $gbest$ according to their corresponding neighborhood, which contributes to a high possibility of escaping from the local optima; 2) a switching learning strategy is designed where the acceleration coefficients are adaptively adjusted at each iteration, and the velocity updating model is adjusted according to the searching state determined by the evolutionary factor; and 3) the performance of the DNSPSO algorithm is comprehensively evaluated on 14 well-known benchmark functions (including the unimodal, multimodal and rotated multimodal cases), and experimental results demonstrate that the proposed DNSPSO algorithm outperforms some existing PSO algorithms in terms of convergence performance and solution quality, especially for complicated multimodal optimization problems.

The remainder of this paper is organized as follows. In Section II, the traditional PSO algorithm and PSO variants are presented. The distance-based dynamic neighborhood and the proposed DNSPSO algorithm are presented in Section III. In Section IV, the parameter setting, benchmark functions, and experiment results are discussed. Finally, conclusions and future work are presented in Section V.

II. PSO Algorithms

A. Traditional PSO Algorithm

The PSO algorithm is a population-based EC approach where each particle's position serves as a potential solution of the optimization problem [15]. In a PSO algorithm [15], each particle searches through the problem space by learning from its experience and cooperating with other particles.

In a D-dimensional problem space, the position of the ith particle at kth iteration is denoted as $x_i(k) = (x_{i1}(k), x_{i2}(k), \cdots, x_{iD}(k))$. The velocity of particle i at kth iteration is represented by $v_i(k) = (v_{i1}(k), v_{i2}(k), \cdots, v_{iD}(k))$. Rules for updating velocity and position of particle i are described as follows [15]:

$$\begin{aligned} v_i(k+1) =& wv_i(k) + c_1r_1(p_i(k) - x_i(k)) \\ &+ c_2r_2(p_g(k) - x_i(k)) \\ x_i(k+1) =& x_i(k) + v_i(k+1) \end{aligned} \tag{1}$$

where p_i and p_g represent the best solution found by particle i and the entire swarm, known as $pbest$ and $gbest$; w is the inertia weight which is a scale factor that controls the influence of the previous velocity on the current one; r_1 and r_2 are two uniformly distributed random numbers sampled from $[0, 1]$; c_1 and c_2 are the cognitive acceleration coefficient and the social acceleration coefficient which push the particle towards $pbest$ and $gbest$, respectively.

B. PSO Variants

In recent years, a great number of variant PSO algorithms have been developed to improve the search ability of the traditional PSO algorithm. Various parameter updating strategies have been introduced to improve the performance of PSO algorithms by adjusting the parameters (e.g., inertia weight, cognitive acceleration coefficient and social acceleration coefficient) [33]. In general, a larger inertia weight will contribute to the global search, and a smaller inertia weight will benefit the local refinement. A linearly decreasing inertia weight PSO (PSO-LDIW) algorithm has been proposed in [34] to improve the search ability of the PSO algorithm. A PSO algorithm with time-varying acceleration coefficients has been put forward in [30] with the aim to effectively control the local and global explorations. The constriction factor has been introduced into the PSO algorithm (PSO-CK) in [6] to enhance the convergence rate and search ability. In particular, a Markov chain has been utilized in the switching PSO (SPSO) algorithm proposed in [38] to adaptively control the inertia weight and acceleration coefficients.

Some variant PSO algorithms focus on designing new topology structures aiming to guarantee the population diversity and prevent premature convergence. Several popular topologies (including all, ring, clusters, pyramid, and von Neumann) have been introduced into PSO algorithms [16]. By making full use of neighborhood information, a fully informed PSO algorithm has been developed in [26]. In the comprehensive learning PSO algorithm, a new learning strategy has been put forward which updates each particle by utilizing all other particles' individual best information [18]. Additionally, a locally informed PSO algorithm has been introduced in [29] by using several local best positions through the neighborhood to guide the search of the particles.

Another research direction in designing new PSO algorithms is to hybridize with other EC algorithms. It has been revealed that the combination of the PSO algorithm and other EC methods shows promise in improving the performance of the PSO algorithm [45]. Additionally, there is a growing research interest on introducing the sociological/biological inspired methods into the PSO algorithm, such as the niching PSO algorithm [5], the aging theory inspired PSO algorithm [7], and the cultural-based PSO algorithm [8], etc.

Recently, the complex system and control theory have been employed into the PSO algorithms, such as chaos, fuzzy, neural networks, time-delay, and the orthogonal learning [10], [20], [28], [35], [47]. As a notable example, the delayed information of the particles has been utilized in the switching delayed PSO (SDPSO) algorithm to make full use of the historical information of the evolution process, thereby facilitating a better exploration of the problem space than the traditional PSO algorithm [38].

III. A Dynamic-Neighborhood-Based Switching PSO Algorithm

In this section, the proposed DNSPSO algorithm is discussed mainly from three aspects: 1) the distance-based dynamic neighborhood is introduced to adjust the $pbest$ and $gbest$ based on their corresponding neighborhood, which benefits the thorough exploration of the problem space so as to alleviate the premature convergence problem; 2) the developed switching strategy is utilized to adaptively choose the acceleration coefficients and update the velocity according to the searching state; and 3) the DE method is employed to enhance the diversity of the PSO algorithm. Therefore, the proposed DNSPSO algorithm could thoroughly search the problem space in order to effectively discover the optimal solution.

A. Framework of the DNSPSO Algorithm

In the proposed DNSPSO algorithm, the velocity and position of particle i at iteration k are updated by the following mechanism:

$$\begin{aligned} v_{i,j}(k+1) =& w(k)v_{i,j}(k) \\ &+ c_1(\xi(k))r_1(pbest^j(\xi(k)) - x_{i,j}(k)) \\ &+ c_2(\xi(k))r_2(gbest^j(\xi(k)) - x_{i,j}(k)), \\ x_{i,j}(k+1) =& x_{i,j}(k) + v_{i,j}(k+1) \end{aligned} \tag{2}$$

where r_1 and r_2 are two random numbers distributed in [0,1]; j denotes the dimension of the problem space; $c_1(\xi(k))$ and $c_2(\xi(k))$ are the cognitive acceleration coefficient and the social acceleration coefficient, respectively.

Notice that $c_1(\xi(k))$, $c_2(\xi(k))$, $pbest^j(\xi(k))$ and $gbest^j(\xi(k))$ are determined according to a mode-dependent Markov chain $\xi(k)$ $(k \geq 0)$. The Markov chain, which is determined by a probability transition matrix $\mathcal{P} = (p_{ij})_{J\times J}$, takes a value in a finite state space: $\mathcal{F} = \{1, 2, \cdots, J\}$ where $p_{ij} \geq 0$ $(i, j \in \mathcal{F})$ and $\sum_{j=1}^{J} p_{ij} = 1$. In this paper, $\xi(k)$ represents four different searching states ($J = 4$) [38].

In particular, the current searching state of each particle is determined by the probability transition matrix $\mathcal{P}$ and an evolutionary factor E_f [46]. E_f is an index for describing the distribution property of swarm:

$$E_f(i) = \frac{d_i - d_{\min}}{d_{\max} - d_{\min}} \tag{3}$$

where $d_{\max}$ and $d_{\min}$ denote the maximum and minimum distances among d_i, respectively. d_i is the mean distance between particle i and other particles, which is calculated as follows:

$$d_i = \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} \sqrt{\sum_{k=1}^{D} (x_i^k - x_j^k)^2} \tag{4}$$

where N and D respectively represent the swarm size and the dimension of the problem space.

It should be mentioned that E_f defined in Eq. 3 is different from it in [46]. To be specific, in this paper, d_i is defined in Eq. 4, which is different from that in [46]. The classification rule of the searching states and the probability transition matrix $\mathcal{P}$ are given as follows:

$$\begin{cases} Convergence: & \xi(k) = 1, & 0 \leq E_f < 0.25 \\ Exploitation: & \xi(k) = 2, & 0.25 \leq E_f < 0.5 \\ Exploration: & \xi(k) = 3, & 0.5 \leq E_f < 0.75 \\ Jumping\ out: & \xi(k) = 4, & 0.75 \leq E_f \leq 1 \end{cases}$$

$$\mathcal{P} = \begin{pmatrix} \pi & 1-\pi & 0 & 0 \\ \frac{1-\pi}{2} & \pi & \frac{1-\pi}{2} & 0 \\ 0 & \frac{1-\pi}{2} & \pi & \frac{1-\pi}{2} \\ 0 & 0 & 1-\pi & \pi \end{pmatrix} \tag{5}$$

where π is the transition probability.

B. Distance-Based Dynamic Neighborhood

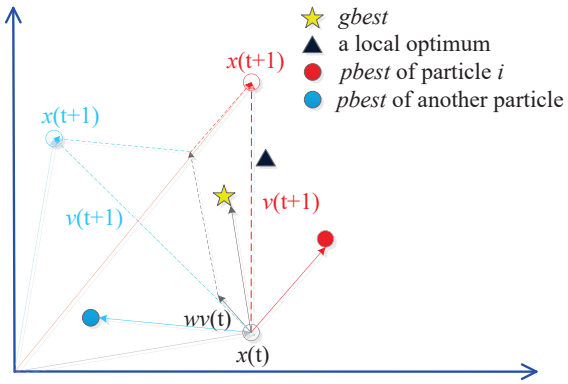

Fig. 1. The schematic diagram of a particle's trajectory in the DNSPSO.

As mentioned previously, the conventional PSO algorithm updates each particle only based on its individual $pbest$ and the $gbest$ discovered by the entire swarm. Fig. 1 shows the shortcoming of the updating strategy of the conventional PSO algorithm, where yellow star stands for $gbest$, black triangle

denotes a local optimum, red circle is $pbest$ of the particle represented by the white circle, and blue circle is $pbest$ of another particle. It should be pointed out that vector addition in Fig. 1 satisfies the parallelogram law. The lengths of some components are slightly changed due to the influence of the acceleration coefficients c_1, c_2 and the random numbers r_1, r_2. According to Eq. 1, the red circle and the yellow star are utilized to determine the next position of the white particle. In this condition, the movement trajectory is the red dashed line which is close to the local optima. Notice that if the red circle is replaced with the blue one, result will be significantly different. Therefore, we can draw a conclusion that making full use of the information of the swarm could contribute to the alleviation of the premature convergence problem, especially for the test functions with a great many of local optimal solutions.

To handle this drawback, in this paper a dynamic neighborhood is introduced to enhance communication in the swarm, and the neighborhood of particle p_i is defined as the nearest k particles by calculating the Euclidean distance between its current $pbest$ and other particles. Through the evolution process, the neighborhood of each particle adjusts automatically at each iteration. To make full use of the neighborhood information of other particles, a new updating mechanism of $pbest$ and $gbest$ is designed in this paper based on the developed dynamic neighborhood. In the proposed distance-based dynamic neighborhood, particle p_i randomly selects another particle among the swarm, and learns from the neighbor of the selected particle's $pbest$. In addition, a randomly selected particle from the neighborhood of current global optimal solution $gbest$ is employed to update the velocity of the particle p_i instead of current $gbest$. By learning from the neighborhood of $gbest$, the possibility of falling into the local optima could be greatly decreased.

The schematic diagram of the proposed learning strategy is shown in Fig. 2, and its detailed procedure is given as follows:

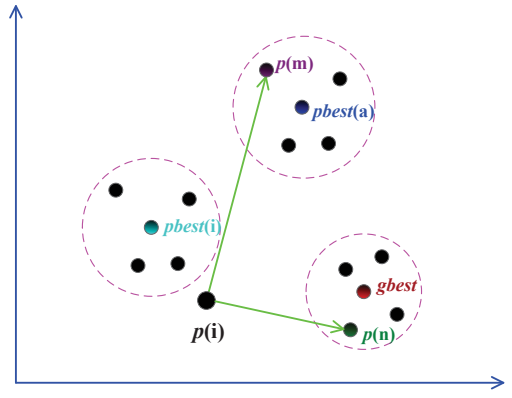

Fig. 2. The schematic diagram of the distance-based dynamic neighborhood, $k = 4$.

Distance-Based Dynamic Neighborhood:
For $i = 1, 2, ..., N$
Identify each particle's neighborhood (i.e., the nearest k particles to $pbest(i)$) based on the calculated Euclidean distance
Identify $gbest$'s neighborhood (the nearest k particles to $gbest$) based on the calculated Euclidean distance
For $j = 1, 2, ..., dim$
Choose one particle from the swarm randomly (except the ith particle), termed as ath particle
Choose one particle from the neighborhood of the $pbest(a)$ randomly, termed as mth particle
Choose one particle from the $gbest$'s neighborhood randomly, termed as nth particle
Learn from the mth particle and the nth particle
Endfor
Endfor

TABLE I
STRATEGIES FOR SELECTING $pbest^j$, $gbest^j$ AND ACCELERATION COEFFICIENTS

State	Mode	c_1	c_2	$pbest^j$	$gbest^j$
Convergence	$\xi(k) = 1$	2	2	$p_{i,j}(k)$	$p_{g,j}(k)$
Exploitation	$\xi(k) = 2$	2.1	1.9	$p_{m,j}(k)$	$p_{g,j}(k)$
Exploration	$\xi(k) = 3$	2.2	1.8	$p_{i,j}(k)$	$p_{n,j}(k)$
Jumping-out	$\xi(k) = 4$	1.8	2.2	$p_{m,j}(k)$	$p_{n,j}(k)$

[1] p_m and p_n represent the particles shown in Fig. 2.

C. Switching Learning Strategy

The switching learning strategy of the proposed DNSPSO algorithm is displayed in Table I. It should be pointed out that c_1 and c_2 are set on the basis of our previous work [43]. Meanwhile, more details can be founded in [46] where the authors have presented a strategy for selecting c_1 and c_2.

According to Eq. 2, a large value of c_1 benefits the individual search and a small value of c_2 mitigates the influence of $gbest$. The main purpose of the exploitation and exploration states is to thoroughly discover the search space and avoid the particles being trapped in local optima. In this case, both exploitation and exploration states have a small value of c_2 and a large value of c_1. In the jumping-out state, a large value of c_2 and a small value of c_1 are adopted, which aims to push the particles away from current best positions.

Except for value settings for acceleration coefficients c_1 and c_2, more comprehensive descriptions of the four searching states are provided as follows:

- In the convergence state, the velocity updating strategy in the DNSPSO algorithm is the same as that of the traditional PSO algorithm. That is, each particle learns from its own $pbest$ and the current $gbest$.
- In the exploitation state, particles are encouraged to search the problem space as much as possible, especially the regions around $pbest$ of each individual particle. For the particle p_i, we first randomly select another particle among the swarm, and then randomly select a particle termed as p_m from the neighbor of the selected particle's $pbest$. Finally, we replace the original $pbest$ of particle p_i by p_m. In this way, the particles share information among the swarm, which benefits the particles to exploit through the whole problem space.

- In the exploration state, all the particles tend to search around the region of discovered optimal solution for preventing from falling into the local optimal. Therefore, we randomly select a particle termed as p_n in the distance-based dynamic neighborhood of $gbest$ to replace the current $gbest$. In this way, the swarm in the DNSPSO algorithm could have wider searching regions instead of crowding around the current $gbest$ with high possibility of being a local optimum.
- In the jumping-out state, all the particles are driven to escape from the current local optimum and discover a better optimal solution. In this case, we simultaneously adjust the $pbest$ and $gbest$ like steps in the exploitation and exploration phases in order to find a better global optimum as soon as possible.

In addition, the evolutionary factor E_f represents the population diversity. The larger the E_f, the worse the diversity. The inertia weight w is determined by E_f, which is shown in

$$w = 0.5 * E_f + 0.4 \tag{6}$$

and more detailed information can be refereed in [38].

D. Differential Evolution

The DE algorithm, as a stochastic population-based evolutionary algorithm, has been extensively investigated for improving the performance of the PSO algorithm [42], [45]. The hybridization of the DE algorithm could not only guarantee the diversity of the $pbest$ and $gbest$ but also enhance the capability of escaping from the local optima. In this paper, the DE algorithm exploits the $pbest$ of all particles with the aim of generating new improved positions according to the following steps.

First, a mutant vector is generated by the mutation operation:

$$v_{i,j} = pbest_{i,j} + F * (pbest_{r_1,j} - pbest_{r_2,j}) \tag{7}$$

where r_1 and r_2 are random numbers which are in the range of $[1, 2, ..., N]$; and F represents the mutation factor $\in [0, 1]$.

Then, a trial vector is replaced with the mutant vector if the randomly generated number r is less than the crossover constant CR, otherwise, it takes value of the current $pbest$:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } r \leq CR \\ pbest_{i,j}, & \text{otherwise} \end{cases} \tag{8}$$

Finally, the $pbest_i$ and $gbest$ are updated by the winner between the generated trial vector and the corresponding original values, in terms of the values of fitness function (taking the minimization problem as an example):

$$\begin{cases} pbest_i = u_i, & \text{if } f(u_i) < f(pbest_i) \\ gbest = u_i, & \text{if } f(u_i) < f(gbest) \end{cases} \tag{9}$$

Hence, the DE algorithm is integrated perfectly into the proposed DNSPSO approach, and the flowchart of the proposed DNSPSO algorithm is illustrated in Fig. 3.

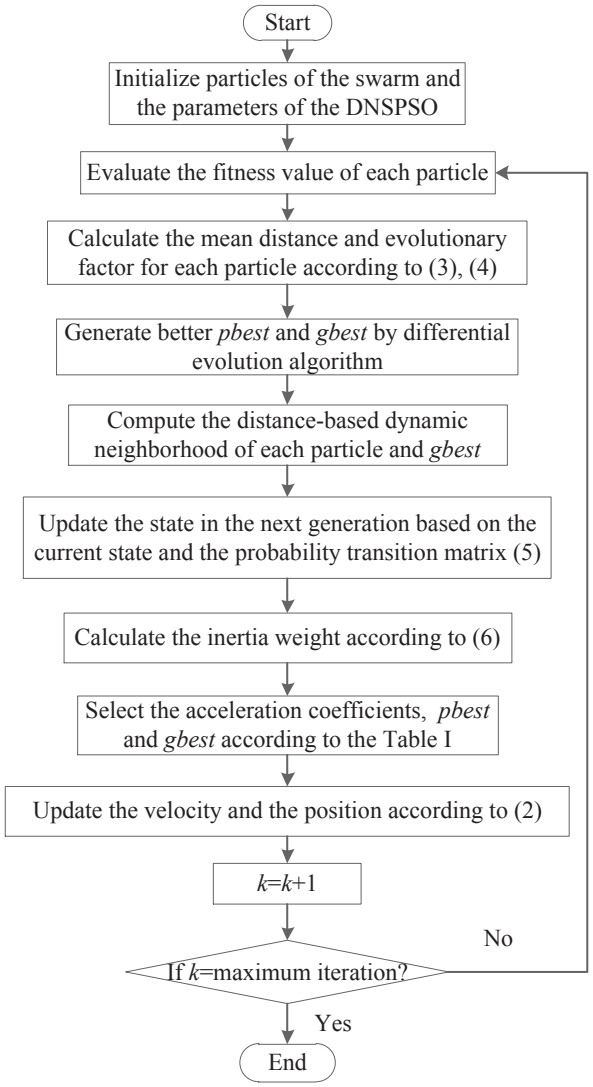

Fig. 3. The flowchart of DNSPSO algorithm

IV. Simulation Experiments

A. Experiments Setup

To verify the validity of the proposed DNSPSO algorithm, experiments are conducted on 14 benchmark functions in terms of unimodal functions, multimodal functions and rotated multimodal functions [31], [37], [41]. It should be pointed out that the developed DNSPSO algorithm aims to improve the performance of PSO on complicated multimodal optimization problems. $f_1(x)$ and $f_2(x)$ are unimodal functions which are chosen as the first group of benchmark functions. The second group is composed of seven complex multimodal functions ($f_3(x) - f_9(x)$) which contain a large number of local optima. The third group includes four rotated multimodal functions, where the rotated variable y is generated by left multiplying an orthogonal matrix M, described as $y = M * x$. Note that the orthogonal matrix is generated by using the Salomon's method [31]. The detailed information of all the benchmark functions is shown in Table II, where "Acceptance" represents the threshold for determining whether solutions found by PSO

TABLE III
SELECTED PSO ALGORITHMS

Algorithm	Parameters	Reference
PSO-LDIW	$w: 0.9-0.4, c_1 = c_2 = 2$	[34]
PSO-TVAC	$w: 0.9-0.4, c_1: 2.5-0.5, c_2: 0.5-2.5$	[30]
PSO-CK	$w: 0.729, c_1 = c_2 = 1.49$	[6]
SPSO	Automatically chosen	[38]
LEPSO	$w: 0.9-0.4, c_1 = c_2 = 2$	[1]
SLEPSO	Automatically chosen	[42]
DNSPSO	Automatically chosen	**Proposed**

algorithms are acceptable or not. If the obtained solution falls within the threshold, it is regarded as a successful result.

The performance of the DNSPSO algorithm is compared with that of 6 existing PSO algorithms including the PSO-LDIW algorithm [34], the PSO-TVAC algorithm [30], the PSO-CK algorithm [6], the SPSO algorithm [38], the LEPSO algorithm [1] and the SLEPSO algorithm [42].

In the proposed DNSPSO algorithm, the transition probability π is set to be 0.9, and the initial state is set as 1 (the convergence state). The size of two neighborhoods ($pbest$ and $gbest$) is 5. In the DE algorithm, the mutation factor F linearly increases from 0.5 to 1, and CR linearly decreases from 0.9 to 0.4. Note that the parameters of the other six PSO algorithms are set according to the corresponding references as shown in Table III.

In our simulation, the swarm contains 30 particles, and the dimension of all benchmark functions is 30. The number of fitness evaluations (FEs) is set to be 5×10^4. Each experiment is repeated 30 times independently for each benchmark function.

B. Comparisons on the Solution Accuracy

In order to comprehensively evaluate the solution quality of the PSO algorithms, the mean value, best value and standard deviation (Std. Dev.) value of each benchmark function are summarized in Table IV, where the best results are highlighted in boldface. For simplicity, the convergence processes of mean fitness values for 7 PSO algorithms applied to 6 typical benchmark functions are graphically shown in Fig. 4.

We can see that the DNSPSO algorithm achieves the best performance on most of 14 test functions among the 7 PSO algorithms. In particular, the DNSPSO algorithm ranks first on 12 functions (f_1, f_3-f_{12}, and f_{14}) according to the mean value, and also ranks second on f_2 and f_{13}.

To be specific, in Table IV, we can see that the SPSO algorithm, the LEPSO algorithm, the SLEPSO algorithm, and the DNSPSO algorithm obtain the global minimum on f_1. In addition, the DNSPSO algorithm gets the second best global mean value which is very close to the best result, with the smallest Std. Dev. value for f_2. Hence, the DNSPSO algorithm demonstrates competitive performance on unimodal problems.

For multimodal problems without rotation, it can be clearly seen that the developed DNSPSO algorithm performs better than other 6 PSO methods. It can be observed from Table IV and Fig. 4 that the solution accuracies are greatly improved by the DNSPSO algorithm when applied to the functions of f_3, f_4, f_6 and f_7. Notice that the DNSPSO algorithm gets the global results f_{min} on f_4 and f_6. In addition, the proposed DNSPSO algorithm achieves the best performance on f_5 and f_8 in terms of the mean value and best value. For f_9, the DNSPSO algorithm, the PSO-TVAC algorithm, the LEPSO algorithm as well as the SLEPSO algorithm all find a satisfactory solution. It is worth mentioning that several PSO algorithms obtain the same mean value and best value on the f_9 function, which reflects the fact that the Penalized problem is easily trapped in the local optimum.

For multimodal problems with rotation, the developed DNSPSO algorithm outperforms the other 6 PSO algorithms. In particular, the DNSPSO algorithm has remarkable performances when solving the functions of f_{11}, f_{12} and f_{14}. That is, only the DNSPSO algorithm can achieve the accuracies of 10^{-17} on f_{11}, 0.8 on f_{12}, and 38 on f_{14}. For f_{10}, the LEPSO, SLEPSO, and DNSPSO algorithms find a close solution, where the results of the SLEPSO and DNSPSO algorithms are slightly better than the result obtained by the LEPSO algorithm. Meanwhile, the proposed DNSPSO algorithm ranks second on f_{13}, which is quite close to the first obtained by the LEPSO algorithm. It should be pointed out that the results of optimization problems with rotation are generally worse than those of the optimization problems without rotation, which is related to the increase of the problem complexity.

C. Comparisons of Convergence Performance

It is known that convergence performance is of key importance to evolutionary algorithms. The mean FEs, successful rate (SR(%)) and success performance (SP) are selected as performance indices [47] for comparison in a quantitative manner, which is shown in Table V. The mean FEs stands for the average evaluation number required to reach an acceptable solution, notably, it only takes the "successful" trails of all 30 runs into account. SR(%) represents the successful percentage of 30 trails for each test function. As some PSO algorithms cannot search the optimal solution in every trail on some functions, the mean FEs cannot comprehensively demonstrate the convergence performance of PSO algorithms. Therefore, SP is exploited to further evaluate the evolutionary performance, where $\mathrm{SP} = \frac{\mathrm{Mean\ FEs}}{\mathrm{SR}}$. In addition, the ranks of SP and SR for the selected PSO algorithms are also presented in Table V. It should be mentioned that the corresponding SP ranking is 7 if there is no successful trail for a function.

It can be clearly seen from Table V that the DNSPSO algorithm finds a solution within the acceptable threshold for all test functions, which is verified by the SR index with 100% on all the benchmark functions. Notice that some PSO algorithms fail all trails on some optimization problems, for example, PSO-CK fails on f_5, f_7, f_{10}, SPSO fails on f_7, f_{10}, and PSO-TVAC fails on f_{10}. The proposed DNSPSO algorithm yields the highest reliability with 100% on average SR, followed by LEPSO (98.57%), SLEPSO (92.14%), PSO-TVAC (85.71%), PSO-LDIW (84.29%), SPSO (60%), and PSO-CK (55%). Therefore, the DNSPSO algorithm is a reliable optimization algorithm with a high success rate on both unimodal and multimodal problems.

In terms of SP, the DNSPSO algorithm on 14 functions ranks second overall among the selected PSO algorithms. The

TABLE II
FOURTEEN BENCHMARK FUNCTIONS USED FOR THE COMPARISON

Test Function	D	Search Space	Global x	Global f_{min}	Acceptance	Name
f_1	30	$[-100, 100]^D$	$\{0\}^D$	0	0.01	Sphere
f_2	30	$[-10, 10]^D$	$\{0\}^D$	0	100	Rosenbrock
f_3	30	$[-5.12, 5.12]^D$	$\{0\}^D$	0	50	Rastrigin
f_4	30	$[-5.12, 5.12]^D$	$\{0\}^D$	0	50	Noncontinuous Rastrigin
f_5	30	$[-32, 32]^D$	$\{0\}^D$	0	0.01	Ackley
f_6	30	$[-600, 600]^D$	$\{0\}^D$	0	0.01	Griewank
f_7	30	$[-500, 500]^D$	$\{420.96\}^D$	0	2000	Schwefel
f_8	30	$[-0.5, 0.5]^D$	$\{0\}^D$	0	0.01	Weierstrass
f_9	30	$[-50, 50]^D$	$\{0\}^D$	0	0.01	Generalized Penalized
f_{10}	30	$[-32, 32]^D$	$\{0\}^D$	0	0.01	Rotated Ackley
f_{11}	30	$[-600, 600]^D$	$\{0\}^D$	0	100	Rotated Griewank
f_{12}	30	$[-0.5, 0.5]^D$	$\{0\}^D$	0	10	Rotated Weierstrass
f_{13}	30	$[-5.12, 5.12]^D$	$\{0\}^D$	0	100	Rotated Rastrigin
f_{14}	30	$[-5.12, 5.12]^D$	$\{0\}^D$	0	100	Rotated Noncontinuous Rastrigin

D represents the dimension of test function.

TABLE IV
THE COMPARISONS OF SEARCHING RESULTS AMONG SEVEN PSO ALGORITHMS ON FOURTEEN BENCHMARK FUNCTIONS

		PSO-TVAC	PSO-LDIW	PSO-CK	SPSO	LEPSO	SLEPSO	**DNSPSO**
f_1	Mean	7.76e-128	1.45e-307	2.50e-323	**0**	**0**	**0**	**0**
	Best value	6.70e-168	3.00e-323	0	**0**	**0**	**0**	**0**
	Std. Dev.	2.45e-127	0	0	**0**	**0**	**0**	**0**
f_2	Mean	20.97	65.84	2.34	26.76	**4.88e-30**	5.02e-30	4.95e-30
	Best value	1.33	12.02	3.52e-21	3.99	0	0	**0**
	Std. Dev.	20.67	99.83	2.66	26.45	1.06e-29	7.75e-30	**7.70e-30**
f_3	Mean	17.01	10.94	62.50	54.13	5.07	30.05	**1.19**
	Best value	11.94	4.97	40.79	33.83	0.99	14.92	**0**
	Std. Dev.	3.88	4.01	16.24	15.82	2.46	9.86	**1.31**
f_4	Mean	13.70	3.30	36.30	14.70	0.30	1	**0**
	Best value	8	0	17	2	0	0	**0**
	Std. Dev.	3.86	7.72	16.56	10.59	0.48	1.41	**0**
f_5	Mean	8.70e-15	7.99e-15	1.84	0.79	7.64e-15	6.93e-15	**6.57e-15**
	Best value	7.99e-15	7.99e-15	0.93	4.44e-15	4.44e-15	4.44e-15	**4.44e-15**
	Std. Dev.	2.25e-15	0	0.55	0.87	**1.12e-15**	1.72e-15	1.83e-15
f_6	Mean	0.0275	0.0135	0.0231	0.0152	7.40e-04	0.0079	**0**
	Best value	3.44e-15	0	0	0	0	0	**0**
	Std. Dev.	0.0284	0.0129	0.0226	0.0148	0.00234	0.0105	**0**
f_7	Mean	2.03e+03	2.41e+03	5.69e+03	5.84e+03	7.92e+02	2.83e+03	**5.45e+02**
	Best value	1.30e+03	1.88e+03	4.80e+03	4.94e+03	1.18e+02	1.34e+03	**3.82e-04**
	Std. Dev.	4.70e+02	5.04e+02	6.11e+02	5.65e+02	4.71e+02	1.17e+03	**4.07e+02**
f_8	Mean	2.42e-14	5.65e-06	2.48	1.69	1.71e-14	1.49e-14	**1.21e-14**
	Best value	2.13e-14	1.42e-14	1.42e-14	0.00713	1.42e-14	0	**7.11e-15**
	Std. Dev.	3.67e-15	1.79e-05	1.75	2.04	**3.67e-15**	7.07e-15	4.80e-15
f_9	Mean	1.57e-32	1.58e-32	0.249	1.57e-32	1.57e-32	1.57e-32	**1.57e-32**
	Best value	1.57e-32	1.57e-32	1.57e-32	2.73e-32	1.57e-32	1.57e-32	**1.57e-32**
	Std. Dev.	2.88e-48	4.08e-34	0.3067	0.265	2.88e-48	2.88e-48	**2.88e-48**
f_{10}	Mean	1.80	1.33	2.97	2.51	7.28e-15	6.93e-15	**6.93e-15**
	Best value	1.34	7.99e-15	1.78	0.93	4.44e-15	4.44e-15	**4.44e-15**
	Std. Dev.	0.40	0.98	1.04	0.87	1.50e-15	1.72e-15	**1.72e-15**
f_{11}	Mean	0.0111	0.0182	0.0186	0.0235	1.66e-04	0.00591	**1.11e-17**
	Best value	0	0	0	0	0	0	**0**
	Std. Dev.	0.0122	0.0233	0.0279	0.0301	5.26e-04	0.00659	**3.51e-17**
f_{12}	Mean	4.59	4.36	10.25	9.99	1.43	3.12	**0.819**
	Best value	2.69	1.58	6.54	6.91	0.199	0.639	**9.04e-04**
	Std. Dev.	1.27	1.31	2.69	2.07	1.07	1.98	**0.60**
f_{13}	Mean	41.73	57.21	1.06e+02	66.07	**30.35**	49.95	38.70
	Best value	22.91	40.79	95.68	42.56	21.89	37.81	**20.89**
	Std. Dev.	15.91	15.98	12.00	15.02	**6.05**	8.15	14.06
f_{14}	Mean	42.00	51.50	74.13	60.81	43.53	59.58	**38.10**
	Best value	**20.00**	37.00	36.13	38.20	28	27	29
	Std. Dev.	25.10	11.77	22.71	15.68	9.77	26.16	**7.56**

DNSPSO algorithm performs the best on f_{11}, f_{13}, and f_{14}, ranks second on 5 functions (f_5, f_7, f_9, f_{10}, f_{12}), ranks third on f_6 and f_8, ranks fourth on f_1 and f_2, and ranks seventh on f_3 and f_4. For two unimodal problems, the DNSPSO algorithm

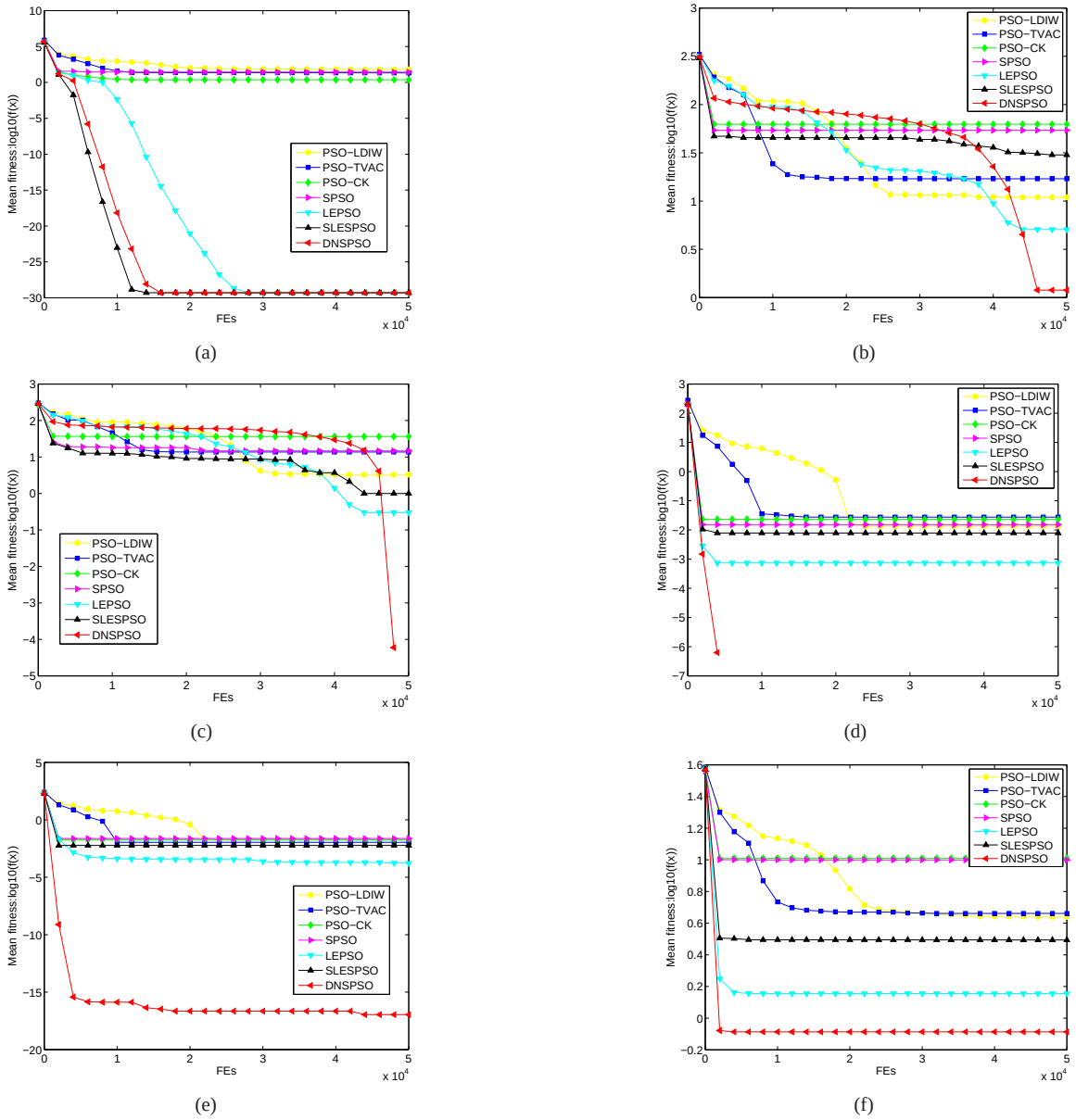

Fig. 4. Convergence performance on benchmark functions. (a) Rosenbrock. (b) Rastrigin. (c) Noncontinuous Rastrigin. (d) Griewank. (e) Rotated Griewank. (f) Rotated Weierstrass.

falls behind three algorithms (PSO-CK, SPSO, and SLEPSO) but with extremely close performance. This phenomenon is consistent with the fact that particles in the DNSPSO algorithm utilize the information of neighborhoods of $pbest$ and $gbest$ to escape from local optima, which leads to a bigger mean FEs than the PSO-CK algorithm, the SPSO algorithm, and the SLEPSO algorithm.

In addition, the DNSPSO algorithm needs smaller mean FEs when applied to optimizing most multimodal functions without rotation. In particular, the DNSPSO algorithm significantly outperforms the other contenders in a landslide by obtaining three first and two second on five multimodal functions with rotation. Meanwhile, the DNSPSO algorithm gets the largest mean FEs on the f_3 and f_4, which may be caused by the characteristics of Rastrigin and Noncontinuous Rastrigin functions containing a large number of local optima. Note that the solution accuracies of f_3 and f_4 obtained by the DNSPSO algorithm greatly surpass the other PSO algorithms, with a rapid decline in the range from 4×10^4 to 5×10^4 FEs, as shown in Fig. 4. In summary, the developed DNSPSO algorithm demonstrates satisfactory convergence rate on both unimodal and multimodal problems, and significantly outperforms other PSO variants on multimodal problems.

D. Analysis on the Neighborhood Size k

The selection of neighborhood size k is of vital importance to the performance of the DNSPSO algorithm. A series of simulation experiments are conducted to find out how the neighborhood size influences the algorithm performance. In this work, two unimodal functions, three multimodal functions without rotation and two multimodal functions with rotation

TABLE V
THE COMPARISONS OF CONVERGENCE RATE AND SUCCESSFUL RATE AMONG SEVEN PSO ALGORITHMS ON FOURTEEN BENCHMARK FUNCTIONS

		PSO-TVAC	PSO-LDIW	PSO-CK	SPSO	LEPSO	SLEPSO	**DNSPSO**
f_1	Mean FEs	8540	20815	336	312	885	**302**	514
	SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	SP	8540	20815	336	312	885	**302**	514
	SP Rank	6	7	3	2	5	1	4
f_2	Mean FEs	7940	19288	218	199	394	**157**	238
	SR(%)	**100**	90	**100**	**100**	**100**	**100**	**100**
	SP	7940	21400	218	199	394	**157**	238
	SP Rank	6	7	3	2	5	1	4
f_3	Mean FEs	8140	18600	**142**	174	17400	10900	31000
	SR(%)	**100**	**100**	20	40	**100**	**100**	**100**
	SP	8140	18600	710	**435**	17400	10900	31000
	SP Rank	3	6	2	1	5	4	7
f_4	Mean FEs	9130	21600	650	**503**	17300	522	26500
	SR(%)	**100**	**100**	90	**100**	**100**	**100**	**100**
	SP	9130	21600	722	**503**	17300	522	26500
	SP Rank	4	6	3	1	5	2	7
f_5	Mean FEs	8960	21400	-	403	4620	**363**	603
	SR(%)	**100**	**100**	0	50	**100**	**100**	**100**
	SP	8960	21400	-	805	4620	**363**	603
	SP Rank	5	6	7	3	4	1	2
f_6	Mean FEs	11200	21600	384	**312**	1400	621	913
	SR(%)	40	50	40	50	**100**	80	**100**
	SP	27900	43200	959	**625**	1400	777	913
	Rank	5	7	4	1	5	2	3
f_7	Mean FEs	**6920**	18400	-	-	29300	44600	27600
	SR(%)	60	20	0	0	**100**	20	**100**
	SP	**11500**	92200	-	-	29300	223000	27600
	SP Rank	1	4	7	4	3	5	2
f_8	Mean FEs	471	4390	871	1879	**18.20**	20	29.60
	SR(%)	**100**	**100**	20	10	**100**	**100**	**100**
	SP	471	4390	4360	18790	**18.20**	20	29.60
	SP Rank	4	6	5	7	1	2	3
f_9	Mean FEs	8420	18700	638	414	2300	**275**	483
	SR(%)	**100**	**100**	40	40	**100**	**100**	**100**
	SP	8420	18700	1590	1035	2300	**275**	483
	SP Rank	6	7	3	4	5	1	2
f_{10}	Mean FEs	-	23895	-	-	2070	**386**	639
	SR(%)	0	20	0	0	**100**	**100**	**100**
	SP	-	119475	-	-	2070	**386**	639
	SP Rank	7	4	7	7	3	1	2
f_{11}	Mean FEs	12.10	9.30	7.70	5.70	8.60	5.50	**4.90**
	SR(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	SP	12.10	9.30	7.70	5.70	8.60	5.50	**4.90**
	SP Rank	7	6	4	3	5	2	1
f_{12}	Mean FEs	6920	16700	196	1330	341	**119**	213
	SR(%)	**100**	**100**	60	60	**100**	**100**	**100**
	SP	6920	16700	327	2210	341	**119**	213
	SP Rank	6	7	3	5	4	1	2
f_{13}	Mean FEs	10900	16500	123	113	15500	183	**101**
	SR(%)	**100**	**100**	**100**	**100**	**100**	90	**100**
	SP	10900	16500	123	113	15500	203	**101**
	SP Rank	5	7	3	2	6	4	1
f_{14}	Mean FEs	8360	15200	126	112	6830	763	**108**
	SR(%)	**100**	**100**	**100**	90	80	**100**	**100**
	SP	8360	15200	126	125	8538	763	**108**
	SP Rank	5	7	3	2	6	4	1
Ave. SP rank		5.07	6.21	4.07	3.36	4.43	2.21	2.93
Final SP rank		6	7	4	3	5	1	2
Ave. SR		85.71%	84.29%	55%	60%	98.57%	92.14%	100%
SR rank		4	5	7	6	2	3	1

are chosen for analyzing the neighborhood size. The results of mean value, Std. Dev, mean FEs and SR(%) are shown in Table VI. It should be pointed out that 1) the DNSPSO algorithm obtains the SR of 100% for all selected seven functions with different k; and 2) the results of SP are the same as the mean FEs and are therefore not shown in Table VI.

For the Sphere optimization problem, the DNSPSO algorithm performs well with different neighborhood sizes, and the DNSPSO algorithm with a smaller k needs a smaller mean FEs. For a complex unimodal problem and the Rosenbrock problem, the performances of the DNSPSO algorithm with different k are inconsistent. The results of the DNSPSO

TABLE VI
INFLUENCES OF DIFFERENT NEIGHBORHOOD SIZES ON SEVEN TYPICAL FUNCTIONS

Test Function	Indicators	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$
Sphere	Mean	**0**	**0**	**0**	**0**	**0**
	Std. Dev.	**0**	**0**	**0**	**0**	**0**
	Mean FEs	**499.2**	504	508.5	514.3	516.3
	SR(%)	**100**	**100**	**100**	**100**	**100**
Rosenbrock	Mean	0.3987	6.6967e-30	**5.2669e-30**	0.3987	0.3987
	Std. Dev.	1.2607	1.4810e-29	**1.2581e-29**	1.2607	1.2607
	Mean FEs	275	**247**	279.8	341.1	262.1
	SR(%)	**100**	**100**	**100**	**100**	**100**
Rastrigin	Mean	0.7960	**0.2985**	0.3980	0.7960	1.2934
	Std. Dev.	0.9143	**0.4806**	0.5138	0.9143	1.3308
	Mean FEs	30365.1	29351.7	**22297.7**	29391.6	28442.7
	SR(%)	**100**	**100**	**100**	**100**	**100**
Griewank	Mean	7.3960e-4	9.8573e-4	**0**	**0**	9.8573e-4
	Std. Dev.	0.0023	0.0031	**0**	**0**	0.0031
	Mean FEs	711.7	836.7	**652**	**652**	689.9
	SR(%)	**100**	**100**	**100**	**100**	**100**
Schwefel	Mean	645.4901	495.4677	**436.2485**	923.8213	992.9087
	Std. Dev.	462.8494	**352.4337**	628.8176	703.7634	495.0306
	Mean FEs	23031	25723.1	29288.6	23205.7	**14687.8**
	SR(%)	**100**	**100**	**100**	**100**	**100**
Rotated Ackley	Mean	**6.9278e-15**	**6.9278e-15**	**6.9278e-15**	**6.9278e-15**	**6.9278e-15**
	Std. Dev.	1.7161e-15	1.835e-15	1.835e-15	1.7161e-15	**1.4980e-15**
	Mean FEs	**622.8**	643.6	638.8	642.3	655.5
	SR(%)	**100**	**100**	**100**	**100**	**100**
Rotated Griewank	Mean	0.0017	3.3307e-17	3.3307e-17	**0**	3.2627e-05
	Std. Dev.	0.0037	7.4934e-17	7.4934e-17	**0**	1.0317e-04
	Mean FEs	5.5	4.5	4.9	5.5	**4.2**
	SR(%)	**100**	**100**	**100**	**100**	**100**

algorithm when $k = 4$ and $k = 5$ are much better than those of other neighbor sizes. For multimodal functions without rotation, the DNSPSO algorithm with $k = 4$ and $k = 5$ demonstrates competitive performance on the Rastrigin and Schwefel functions, and it can provide outstanding performances on the Griewank when k is 5 and 6. Furthermore, the DNSPSO algorithm with different neighborhood sizes obtains the same result on the rotated Ackley function. In particular, the DNSPSO algorithm achieves the global optimum on the rotated Griewank problem when the neighborhood size k is set to 6.

Based on above discussions, therefore, we can summarize the effects of neighborhood size k on the performance of the DNSPSO algorithm, which are: 1) the influence of neighborhood size k on the performance of DNSPSO algorithm depends on the characteristic of optimization problems. For the unimodal optimization problems, the neighborhood size can be chosen as a small value. A large neighborhood size is suitable for multimodal problems, especially the rotated multimodal ones; and 2) the neighborhood size is related to the swarm size. In this paper, the neighborhood size is set as $1/6$ of the swarm size.

V. CONCLUSIONS

In this paper, the DNSPSO algorithm has been proposed to improve the search capability of the traditional PSO algorithm. A distance-based dynamic neighborhood has been developed where the neighborhood of the particles is dynamically updated at each iteration. The acceleration coefficients, personal best particle and the global best particle of the DNSPSO algorithm are automatically adjusted via the switching learning strategy depending on the searching state. Particles in the DNSPSO algorithm share information through the developed distance-based dynamic neighborhood where the personal best particle and the global best particle are replaced by the randomly selected particles in their corresponding neighborhoods. Furthermore, the DE algorithm has been utilized to further expand the search space of the particles. Experimental results have shown that the DNSPSO algorithm outperforms six PSO variants on 14 widely used benchmark functions in terms of the solution quality and convergence performance, especially for complicated multimodal optimization problems. The influence of neighborhood size on the performance of the DNSPSO algorithm has also been comprehensively investigated and demonstrates that the selection of neighborhood size should be determined based on the property of the optimization problems and the swarm size. In the future, we aim to further improve the convergence rate of the developed DNSPSO algorithm. We will also apply the DNSPSO algorithm to other research fields such as event-triggered state estimation [9], moving horizon estimation [50], [51], [52] and the self-organizing RBF neural network [12].

REFERENCES

[1] A. Abdullah, S. Deris, S. Hashim, M. Mohamad, and S. Arjunan, An improved local best searching in particle swarm optimization using differential evolution, In: *Proceedings of the 11th International Conference on Hybrid Intelligent Systems*, Melacca, Malaysia, 2011, pp. 115-120.

[2] F. Bergh and A. Engelbrecht, A cooperative approach to particle swarm optimization, *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225-239, 2004.

[3] T. Blackwell and J. Kennedy, Impact of communication topology in particle swarm optimization, *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 4, pp. 689-702, 2019.

[4] M. R. Bonyadi, A theoretical guideline for designing an effective adaptive particle swarm, *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 57-68, 2020.

[5] R. Brits, A. Engelbrecht, and F. van den Bergh, Locating multiple optima using particle swarm optimization, *Applied Mathematics and Computation*, vol. 189, no. 2, pp. 1859-1883, 2007.

[6] M. Clerc and J. Kennedy, The particle swarm: explosion, stability, and convergence in a multi-dimensional complex space, *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58-73, 2002.

[7] W. Chen, J. Zhang, Y. Lin, and E. Chen, Particle swarm optimization with an aging leader and challengers, *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 2, pp. 241-258, 2013.

[8] M. Daneshyari and G. Yen, Cultural-based multiobjective particle swarm optimization, *IEEE Transactions on Cybernetics*, vol. 41, no. 2, pp. 553-567, 2011.

[9] D. Ding, Z. Wang, and Q.-L. Han, A scalable algorithm for event-triggered state estimation with unknown parameters and switching topologies over sensor networks, *IEEE Transactions on Cybernetics*, in press, DOI:10.1109/TCYB.2019.2917543.

[10] H. Gao and W. Xu, A new particle swarm algorithm and its globally convergent modifications, *IEEE Transactions on Cybernetics*, vol. 41, no. 5, pp. 1334-1351, 2011.

[11] H. Han, W. Lu, L. Zhang, and J. Qiao, Adaptive gradient multiobjective particle swarm optimization, *IEEE Transactions on Cybernetics*, vol. 48, no. 11, pp. 3067-3079, 2017.

[12] H. Han, X. Wu, L. Zhang, Y. Tian, and J. Qiao, Self-organizing RBF neural network using an adaptive gradient multiobjective particle swarm optimization, *IEEE Transactions on Cybernetics*, vol. 49, no. 1, pp. 69-82, 2017.

[13] W. Hu, G. Yen, and G. Luo, Many-objective particle swarm optimization using two-stage strategy and parallel cell coordinate system, *IEEE Transactions on Cybernetics*, vol. 47, no. 6, pp. 1446-1459, 2017.

[14] C. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, *IEEE Transactions on System, Man, and Cybernetics-B*, vol. 34, no. 2, pp. 997-1006, 2004.

[15] J. Kennedy and R. Eberhart, Particle swarm optimization, In: *Proceedings of the IEEE International Conference On Neural Network*, Perth, Australia, 1995, pp. 1942-1948.

[16] J. Kennedy and R. Mendes, Population structure and particle swarm performance, In: *Proceedings of the 2002 IEEE Congress on Evolutionary Computation*, Honolulu, USA, 2002, pp. 1671-1676.

[17] J. Liang and P. Suganthan, Dynamic multi-swarm particle swarm optimizer, In: *Proceedings of the 2005 IEEE Swarm Intelligence Symposium*, Pasadena, CA, USA, 2005, pp. 124-129.

[18] J. Liang, A. Qin, P. Suganthan, and S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281-295, 2006.

[19] Y. Lin, Y.-S. Jiang, Y.-J. Gong, Z.-H. Zhan, and J. Zhang, A discrete multiobjective particle swarm optimizer for automated assembly of parallel cognitive diagnosis tests, *IEEE Transactions on Cybernetics*, vol. 49, no. 7, pp. 2792-2805, 2018.

[20] B. Liu, L. Wang, Y. Jin, F. Tang, and D. Huang, Improved particle swarm optimization combined with chaos, *Chaos, Solitons, Fractals*, vol. 25, no. 5, pp. 1261-1271, 2005.

[21] W. Liu, Z. Wang, X. Liu, N. Zeng, and D. Bell, A novel particle swarm optimization approach for patient clustering from emergency departments, *IEEE Transactions on Evolutionary Computation*, vol 23, no. 4, pp. 632-644, 2019.

[22] W. Liu, Z. Wang, Y. Yuan, N. Zeng, K. Hone, and X. Liu, A novel sigmoid-function-based adaptive weighted particle swarm optimizer, *IEEE Transactions on Cybernetics*, in press, DOI: 10.1109/TCYB.2019.2925015.

[23] X.-F. Liu, Z.-H. Zhan, Y. Gao, J. Zhang, S. Kwong, and J. Zhang, Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization, *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 4, pp. 587-602, 2018.

[24] Y. Liu, Q. Cheng, Y. Gan, Y. Wang, Z. Li, and J. Zhao, Multi-objective optimization of energy consumption in crude oil pipeline transportation system operation based on exergy loss analysis, *Neurocomputing*, vol. 332, pp. 100-110, 2019.

[25] Y. Liu, S. Chen, B. Guan, and P. Xu, Layout optimization of large-scale oil-gas gathering system based on combined optimization strategy, *Neurocomputing*, vol. 332, pp. 159-183, 2019.

[26] R. Mendes, J. Kennedy, and J. Neves, The fully informed particle swarm: simpler, maybe better, *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204-210, 2004.

[27] K. Mistry, L. Zhang, S. Neoh, C. Lim, and B. Fielding, A micro-GA embedded PSO feature selection approach to intelligent facial emotion recognition, *IEEE Transactions on Cybernetics*, vol. 47, no. 6, pp. 1496-1509, 2017.

[28] Y. Pehlivanoglu, A new particle swarm optimization method enhanced with a periodic mutation strategy and neural networks, *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 436-452, 2013.

[29] B. Qu, P. Suganthan, and S. Das, A distance-based locally informed particle swarm model for multimodal optimization, *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 387-402, 2013.

[30] A. Ratnaweera, S. Halgamure, H. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 240-55, 2004.

[31] R. Salomon, Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions, *BioSystems*, vol. 39, pp. 263-278, 1996.

[32] P. Shelokar, P. Siarry, V. Jayaraman, and B. Kulkarni, Particle swarm and ant colony algorithms hybridized for improved continuous optimization, *Applied Mathematics and Computation*, vol. 188, no. 1, pp. 129-142, 2007.

[33] Y. Shi and R. Eberhart, Parameter selection in particle swarm optimization, *Evolutionary Programming VII*, Berlin, Germany: Springer, pp. 591-600, 1998.

[34] Y. Shi and R. Eberhart, Empirical study of particle swarm optimization, In: *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, Washington, DC, USA, 1999, pp. 1945-1950.

[35] Y. Shi and R. Eberhart, Fuzzy adaptive particle swarm optimization, In: *Proceedings of the 2001 IEEE Congress on Evolutionary Computation*, Seoul, South Korea, 2001, pp. 101-106.

[36] B. Song, Z. Wang, and L. Zou, On global smooth path planning for mobile robots using a novel multimodal delayed PSO algorithm, *Cognitive Computation*, vol. 9, no. 1, pp. 5-17, 2017.

[37] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, and S. Tiwari, Problem definitions and evaluation criteria for the CEC2005 special session on real-parameter optimization, In: *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, Edinburgh, UK, 2005, pp. 1-50.

[38] Y. Tang, Z. Wang, and J. Fang, Parameters identification of unknown delayed genetic regulatory networks by a switching particle swarm optimization algorithm, *Expert Systems with Applications*, vol. 38, pp. 2523-2535, 2011.

[39] H. Wang, Y. Jin, and J. Doherty, Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems, *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2664-2677, 2017.

[40] Z.-J. Wang, Z.-H. Zhan, Y. Lin, W.-J. Yu, H. Wang, S. Kwong, and J. Zhang, Automatic niching differential evolution with contour prediction approach for multimodal optimization problems, *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 114-128, 2020.

[41] X. Yao, Y. Liu, and G. Lin, Evolutionary programming made faster, *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82-102, 1999.

[42] N. Zeng, Y. S. Hung, Y. Li, and M. Du, A novel switching local evolutionary PSO for quantitative analysis of lateral flow immunoassay, *Expert Systems with Application*, vol. 41, no. 4, pp. 1708-1715, 2014.

[43] N. Zeng, Z. Wang, H. Zhang, and F. E. Alsaadi, A novel switching delayed PSO algorithm for estimating unknown parameters of lateral flow immunoassay, *Cognitive Computation*, vol. 8, no. 2, pp. 143-152, 2016.

[44] N. Zeng, Z. Wang, H. Zhang, K. Kim, Y. Li, and X. Liu, An improved particle filter with a novel hybrid proposal distribution for quantitative analysis of gold immunochromatographic strips, *IEEE Transactions on Nanotechnology*, vol. 18, no. 1, pp. 819-829, 2019.

[45] N. Zeng, H. Zhang, Y. Chen, B. Chen, and Y. Liu, Path planning for intelligent robot based on switching local evolutionary PSO algorithm, *Assembly Automation*, vol. 36, no. 2, pp. 120-126, 2016.

[46] Z. Zhan, J. Zhang, Y. Li, and H. Chung, Adaptive particle swarm optimization, *IEEE Transactions on System, Man and Cybernetics-B*, vol. 39, no. 6, pp. 1362-1381, 2009.

[47] Z. Zhan, J. Zhang, Y. Li, and Y. Shi, Orthogonal learning particle swarm optimization, *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 6, pp. 832-847, 2011.

[48] Y. Zhang and H. Chiang, A novel consensus-based particle swarm optimization-assisted trust-tech methodology for large-scale global optimization, *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2717-2729, 2017.
[49] Q. Zhu, Q. Lin, W. Chen, K. C. Wong, C. A. C. Coello, J. Li, J. Chen, and J. Zhang, An external archive-guided multiobjective particle swarm optimization algorithm, *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2717-2729, 2017.
[50] L. Zou, Z. Wang, J. Hu and D. Zhou, Moving horizon estimation with unknown inputs under dynamic quantization effects, *IEEE Transactions on Automatic Control*, In press, DOI: 10.1109/TAC.2020.2968975.
[51] L. Zou, Z. Wang, Q. Han and D. Zhou, Moving horizon estimation for networked time-delay systems under Round-Robin protocol, *IEEE Transactions on Automatic Control*, vol. 64, no. 12, pp. 5191-5198, Dec. 2019.
[52] L. Zou, Z. Wang, Q. Han and D. Zhou, Moving horizon estimation of networked nonlinear systems with random access protocol, *IEEE Transactions on Systems, Man, and Cyberneics-Systems*, DOI: 10.1109/TSMC.2019.2918002.

Nianyin Zeng was born in Fujian Province, China, in 1986. He received the B.Eng. degree in electrical engineering and automation in 2008 and the Ph. D. degree in electrical engineering in 2013, both from Fuzhou University. From October 2012 to March 2013, he was a RA in the Department of Electrical and Electronic Engineering, the University of Hong Kong. From September 2017 to August 2018, he as an ISEF Fellow founded by the Korea Foundation for Advance Studies and also a Visiting Professor at the Korea Advance Institute of Science and Technology.

Currently, he is an Associate Professor with the Department of Instrumental & Electrical Engineering of Xiamen University. His current research interests include intelligent data analysis, computational intelligent, time-series modeling and applications. He is the author or co-author of several technical papers and also a very active reviewer for many international journals and conferences.

Dr. Zeng is currently serving as Associate Editors for Neurocomputing, Evolutionary Intelligence, and Frontiers in Medical Technology, and also Editorial Board members for Computers in Biology and Medicine, Biomedical Engineering Online, and Mathematical Problems in Engineering.

Zidong Wang (SM'03-F'14) was born in Jiangsu, China, in 1966. He received the B.Sc. degree in mathematics in 1986 from Suzhou University, Suzhou, China, and the M.Sc. degree in applied mathematics in 1990 and the Ph.D. degree in electrical engineering in 1994, both from Nanjing University of Science and Technology, Nanjing, China.

He is currently Professor of Dynamical Systems and Computing in the Department of Computer Science, Brunel University London, U.K. From 1990 to 2002, he held teaching and research appointments in universities in China, Germany and the UK. Prof. Wang's research interests include dynamical systems, signal processing, bioinformatics, control theory and applications. He has published 250+ papers in IEEE Transactions and 60+ papers in Automatica. He is a holder of the Alexander von Humboldt Research Fellowship of Germany, the JSPS Research Fellowship of Japan, William Mong Visiting Research Fellowship of Hong Kong.

Prof. Wang serves (or has served) as the Editor-in-Chief for *Neurocomputing*, the Deputy Editor-in-Chief for *International Journal of Systems Science*, and an Associate Editor for 12 international journals including IEEE Transactions on Automatic Control, IEEE Transactions on Control Systems Technology, IEEE Transactions on Neural Networks, IEEE Transactions on Signal Processing, and IEEE Transactions on Systems, Man, and Cybernetics-Part C. He is a Member of the Academia Europaea, a Fellow of the IEEE, a Fellow of the Royal Statistical Society and a member of program committee for many international conferences.

Weibo Liu received the B.S. degree in electrical engineering from the Department of Electrical Engineering & Electronics, University of Liverpool, Liverpool, UK, in 2015, and the Ph.D. degree in computer science from Brunel University London, Uxbridge, UK, in 2019. He is currently a research fellow with the Department of Computer Science at Brunel University London, Uxbridge, UK. His research interests include big data analysis and deep learning techniques.

Hong Zhang received her Bachelor's degree in electrical engineering and automation from the Department of Mechanical & Electrical Engineering, Xiamen University, Xiamen, China, in 2015. She is currently pursuing the Master's degree in Electrical Testing Technology and Instruments at Xiamen University, Xiamen, China. Her research interests include image processing and deep learning techniques.

Kate Hone received the B.A. degree in Experimental Psychology from the University of Oxford, U.K. and the M.Sc. Degree in Work Design and Ergonomics and the Ph.D. degree from the University of Birmingham, U.K.

Professor Hone is Head of the Department of Computer Science at Brunel University London, U.K. She previously held academic posts at the University of Nottingham, U.K. before joining Brunel in 2000. At Brunel she has held a number of posts including Director of the Graduate School between 2009 and 2018.

Professor Hone's research interests include spoken dialogue systems, affective computing, social signals processing, health informatics and intelligent data analysis.

Xiaohui Liu received the B.Eng. degree in computing from Hohai University, Nanjing, China, in 1982 and the Ph.D. degree in computer science from Heriot-Watt University, Edinburg, U.K., in 1988.

He is a Professor of Computing with Brunel University London, Uxbridge, U.K., where he directs the Centre for Intelligent Data Analysis. He has over 100 journal publications in computational intelligence and data science. Prof. Liu was a recipient of the Highly Cited Researchers Award by Thomson Reuters.