COMPLEX SYSTEM MODELING AND SIMULATION
ISSN 2096-9929 03/06 pp 307–326
Volume 3, Number 4, December 2023
DOI: 10.23919/CSMS.2023.0015

Evolutionary Experience-Driven Particle Swarm Optimization with Dynamic Searching

Wei Li, Jianghui Jing, Yangtao Chen, Xunjun Chen*, and Ata Jahangir Moshayedi

Abstract: Particle swarm optimization (PSO) algorithms have been successfully used for various complex optimization problems. However, balancing the diversity and convergence is still a problem that requires continuous research. Therefore, an evolutionary experience-driven particle swarm optimization with dynamic searching (EEDSPSO) is proposed in this paper. For purpose of extracting the effective information during population evolution, an adaptive framework of evolutionary experience is presented. And based on this framework, an experience-based neighborhood topology adjustment (ENT) is used to control the size of the neighborhood range, thereby effectively keeping the diversity of population. Meanwhile, experience-based elite archive mechanism (EEA) adjusts the weights of elite particles in the late evolutionary stage, thus enhancing the convergence of the algorithm. In addition, a Gaussian crisscross learning strategy (GCL) adopts cross-learning method to further balance the diversity and convergence. Finally, extensive experiments use the CEC2013 and CEC2017. The experiment results show that EEDSPSO outperforms current excellent PSO variants.

Key words: particle swarm optimization; experience-based topology structure; elite archive; Gaussian crisscross learning

1 Introduction

Population intelligence algorithm[1] encompasses a class of heuristic search algorithms characterized by their capacity to optimize complex problems without an excessive reliance on algorithmic organizational information. These algorithms exhibit broad applicability in the realm of optimization and computation. Among many population intelligence algorithms, particle swarm optimization[2, 3] has simple form, strong robustness, and fast convergence compared with other population intelligence algorithms (such as artificial bee colony algorithm[4, 5], differential evolution algorithm[6, 7], etc.). It is considered as an excellent candidate algorithm for solving many practical application problems. Therefore, it has been utilized in a wide variety of scientific and industrial applications[8, 9].

Although the particle swarm optimization (PSO) algorithm can perform excellently in solving some complex problems, it is also usually difficult to escape from the local optimum trap, which causes the accuracy of the solution to decrease. Furthermore, complex optimization problems such as power dispatching[10, 11] and fault diagnosis[12, 13] have high requirements on solution effectiveness of algorithm. For such problems, although PSO algorithm possesses a high speed of convergence, it is very challenging to locate a correct solution fast during a restricted time. In fact, the essential reason for this problem lies in the difficult balance between the diversity of algorithms and convergence.

• Wei Li, Jianghui Jing, Yangtao Chen, Xunjun Chen, and Ata Jahangir Moshayedi are with the School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China. E-mail: liwei@jxust.edu.cn; jingjianghui@mail.jxust.edu.cn; chenyangtao@jxust.edu.cn; cxj@jxust.edu.cn; ajm@jxust.edu.cn.
• Ata Jahangir Moshayedi is also with the Khomeini Shahr Branch, Islamic Azad University, Isfahan 86145-311, Iran.
* To whom correspondence should be addressed.
※ This article was recommended by Associate Editor Wenyin Gong.
Manuscript received: 2023-04-11; revised: 2023-06-25; accepted: 2023-07-05

In order to tackle the challenges inherent in PSO algorithms, numerous researchers have undertaken efforts to enhance the performance of particle swarm optimization primarily by focusing on three key aspects.

The first direction is the parameter setting adjustment. The initial PSO algorithm proposed by Eberhart and Kennedy[2] in 1995 did not have inertia weights. To enhance the performance of the algorithm, Gao et al.[14] added new parameter inertia weights to the initial version of the PSO velocity update formulation and proposed the standard particle swarm algorithm. Shi and Eberhart[15] verified the effect of the variation of inertia weights on the performance of the particle swarm algorithm and set the range of inertia weights within the interval [0.4, 0.9] to enhance the performance of the PSO algorithm. Li and Cheng[16] presented a parameter adjustment method based on particle adaptation, which can effectively boost the convergence speed and solution quality of the algorithm. Karimi-Nasab et al.[17] improved the initial version of the PSO algorithm by introducing adaptive parameters to improve the efficiency and convergence speed of the algorithm.

The second direction is the neighborhood topology. The topology of a particle swarm establishes a measure of how well its members are connected to other members. It basically describes the subset of particles with which a particle can exchange information[18]. Kennedy et al.[19] were the first to put the global topology, which is the definition of the neighborhood of a particle in the PSO algorithm as the rest of the particles except itself. The global topology is not the definition of the local topology as the two particles closest to itself[19]. With these two proposed topologies, researchers have found that PSO algorithm with a good topology outperforms the standard PSO algorithm. Shi et al.[20] proposed a hybrid cellular automata mechanism, which used three different lattice structures as neighborhoods to allow particles to interact within the swarm. Many topologies of PSO have been proposed by researchers, and some of the superior PSO topologies are random topology, von Neumann topology, star topology, and ring topology.

The third direction is the combination of other algorithmic strategies. This direction mainly combines the inherent social and cooperative features of the algorithms with other optimization strategies. This optimization strategies originate from different evolutionary paradigms, but all aim at achieving intelligent exploration development. This helps to compensate for weaknesses in the PSO algorithm and can be used to guide the algorithm in purposeful search. Shi et al.[21] proposed the idea of exchanging the most suitable particles between genetic algorithm (GA) and PSO, running both algorithms in parallel at a fixed number of iterations. In the work of the hybrid PSO-GA based evolutionary algorithm, a two-stage mechanism is used for the evolutionary strategy of the particles, where the PSO algorithm is responsible for the evolutionary process, while diversity is maintained by using a GA. The authors used this approach to optimize three unconstrained and three constrained problems with good results[22].

For the three aspects of PSO algorithm improvement, parameter settings are basic and very important. Only suitable parameter values can correctly guide the topology of the population and achieve better performance. Topology is the core framework of the algorithm, and excellent neighborhood topology can design excellent algorithms. Algorithm hybridization is a popular direction, learning the excellent strategies of other algorithms and further enhancing the performance of algorithms on the basis of the original. With the above introduction, the use of parameter setting, neighborhood topology, and algorithm hybridization can increase the diversity and convergence of PSO algorithms. However, facing increasingly complex problems, it is extremely necessary to further improve the performance of the algorithm. In this paper, an evolutionary experience-driven particle swarm optimization with dynamic searching (EEDSPSO) is introduced. EEDSPSO uses an evolutionary experience-driven framework to adaptively update the neighborhood structure of particles as well as the archive content. Thus, more suitable particles are selected to guide the population update. In addition, a Gaussian crisscross learning strategy is employed to adjust, and the weight of different search methods is adjusted throughout evolutionary progress to balance the algorithm's performance requirements at different stages. The main research can be concluded in the following.

(1) Evolutionary experience is introduced as a new driving framework to efficiently utilize the population's empirical information in the evolutionary process.

(2) Rationalize the use of evolutionary experience to update the neighborhood structure of particles as well as the adjustment of the archive content, so as to guide the population to search for more suitable localizations.

(3) A Gaussian crisscross learning strategy is proposed to keep a better balance of diversity and convergence.

The remainder of this paper is presented in the following. Section 2 describes related work for proposed algorithm. Section 3 elaborates the proposed algorithm through framework and strategies. Section 4 demonstrates EEDSPSO's effectiveness and high performance through experiments. And finally Section 5 concludes the work and looks to the future.

2 Related Work

2.1 Traditional PSO

In the traditional PSO algorithm, each particle denotes the corresponding problem's solution. During the algorithm's search process, the particle adjusts its motion direction and step size through the velocity term, so as to achieve the purpose of searching for the optimal solution. In each iteration, the position vector of the particle is $x_i = [x_i^1, x_i^2, x_i^3, ..., x_i^D]$, and the corresponding velocity vector is $v_i = [v_i^1, v_i^2, v_i^3, ..., v_i^D]$. In addition, $\mathrm{Pb}_i = [\mathrm{Pb}_i^1, \mathrm{Pb}_i^2, \mathrm{Pb}_i^3, ..., \mathrm{Pb}_i^D]$ are particle i's historical optimal value. $G = [G^1, G^2, G^3, ..., G^D]$ are population's historical optimal value, where D represents the problem's dimension. The traditional PSO is given as follows:

$$v_i(t) = \omega \cdot v_i(t-1) + c_1 \cdot r_1 \cdot (\mathrm{Pb}_i - x_i(t-1)) + c_2 \cdot r_2 \cdot (G - x_i(t-1)) \quad (1)$$

$$x_i(t) = x_i(t-1) + v_i(t-1) \quad (2)$$

where ω is used to change the previous generation velocity's weight, called the inertia weight. t is the current number of iterations. r_1 and r_2 are the random numbers generated in the interval [0, 1]. c_1 and c_2 are acceleration coefficients adjusting for individual cognitive and social cognitive weights, respectively.

2.2 Neighborhood topology

In the iterative process of populations, the neighborhood topology plays an extremely important tool to facilitate the exchange of information between populations. Only a proper neighborhood topology can guide particles to perform a correct search in the space. Recently, numerous scholars have studied this issue. Qu et al.[23] introduced a distance-based dynamic neighborhood topology. The algorithm's fine search capability is enhanced by dynamically changing the neighborhood size using distance as an indicator. This neighborhood topology can be calculated as follows:

$$V_i^d(t) = \omega \cdot \left(V_i^d(t-1) + \varphi\left(P_i^d - X_i^d(t-1)\right)\right) \quad (3)$$

$$P_i = \frac{\sum_{j=1}^{\mathrm{nsize}} (\varphi_j \cdot \mathrm{nbest}_j)/\mathrm{nsize}}{\varphi} \quad (4)$$

where φ_j and φ are distributed random numbers. nbest_j is the j-th nearest neighborhood to the i-th particle's pbest. nsize is the neighborhood size. This is a classic neighborhood topology. This neighborhood topology makes full use of neighborhood information and increases the algorithm's ability for local search and fine-tuning. Since then, more researchers have proposed more excellent neighborhood topologies. Wang et al.[24] presented a dynamic tournament topology method and achieved good performance in artificial neural networks' optimization. The fuzzy rule based neighborhood method proposed by He et al.[25] used fuzzy rules and individual distribution to generate subpopulations and achieved good results in solving nonlinear systems of equations. Li et al.[26] used a complex network topology based on fitness distance to effectively balance algorithm's diversity and convergence. Zhang et al.[27] used the Voronoi-based neighborhood concept and eliminated the need for additional parameters for the algorithm by increasing the computational complexity. Li et al.[28] introduced a differential evolution algorithm based on the neighborhood strategy, used the fitness distance to base the problem difficulty, and achieved excellent results on a single-objective problem. Because the neighborhood topology method has a decisive influence on the search direction and search effort of the particles, the continued exploration of the neighborhood topology method is extremely necessary.

2.3 Archive mechanism

The archival mechanism has garnered considerable scrutiny as an ancillary strategy for enhancing optimization algorithms. Lin et al.[29] undertook a comprehensive examination by integrating collaborative archives with learning probabilities. The integration of collaborative archives furnishes populations with more auspicious information, thereby facilitating their application in the realm of optimal radar system design[29]. This archive mechanism can be calculated as follows:

$$\mathrm{arch}_\beta^d = r^d \cdot \mathrm{arch}_\beta^d + \left(1 - r^d\right) \cdot \mathrm{pbest}_i^d \quad (5)$$

where r^d is a random number. β is the index of the

worst archive particle to be updated. arch_{β}^{d} is the d-th dimension of the β-th archive particle. This archive mechanism is an excellent method. This method uses archives to gather useful information that leads populations to evolve better. There are more excellent algorithms that use the archive mechanisms. A three-archive strategy (stored elites, profiteers, and outstanding exemplars) was proposed by Xia et al.[30] with excellent performance in different optimization functions. Wei et al.[31] archived elite particles for generating new solutions different from the current population and guiding the population to explore in more promising directions. Pan et al.[32] stored particles satisfying the accuracy requirements in the archive and randomly initialized the surrounding particles. The effect is to prevent the population from being trapped in local optima and increase the population's diversity[32]. Tao et al.[33] used an archival strategy with an exploratory function and an integrated learning approach to generate samples and prevent the population from converging prematurely. External archiving is worth further investigation as an effective method to increase the algorithm's diversity and convergence.

3 Proposed EEDSPSO

3.1 Motivation

PSO has been well developed recently as an excellent method for solving complex problem models. Especially, the improvement based on the neighborhood topology has received wide attention. For traditional neighborhood topology, fixed or dynamic topologies based on distance and the number of stops are often used. These structural approaches tend to ignore the favorable information during the evolution of particles. Populations have different need for diversity and convergence in different evolutionary periods. Neighborhood structures with larger ranges tend to accelerate the convergence of populations, and those with smaller ranges tend to increase the diversity of populations. So, it is extremely necessary for the adjustment of the neighborhood structure. Intuitively, particles with the above-mentioned neighborhood topology cannot select the right particles for learning in the evolutionary process. Thus, it is hard to balance the population's diversity and convergence, leading to the population trapped in local optimum.

In response to the above analysis, breaking the traditional method of dynamically adjusting the topology based on distance or the number of stagnation generations, a new evolutionary experience based driving framework is used. The neighborhood structure of the population and the particles in the elite archive is dynamically updated. It is combined with a Gaussian crisscross learning strategy, thus meeting the needs of the population at different evolutionary stages.

3.2 Framework of EEDSPSO

In this section, two strategies driven by evolutionary experience and a Gaussian crisscross learning strategy are combined. Neighborhood techniques and archival strategies utilize evolutionary experience to construct the main framework of the algorithm. Then, the particles adaptively form their own neighborhoods, and the experience-based elite archive mechanism is used to guide the population update. Finally, a Gaussian crisscross learning strategy further balances the diversity and convergence of the algorithm. A transformation strategy is used to transform between the evolutionary experience-driven adaptive framework and the Gaussian crisscross learning strategy.

The main framework of EEDSPSO is shown in Algorithm 1. From Lines 1−4, the algorithm focuses on the setting of relevant parameters, such as population size, neighborhood size, and archive content.

From Lines 5−11, the particles are dynamically updated using the transformation parameter time to select different population evolution methods. The transformation parameter time can be described as follows:

$$\text{time} = \cos(1/2 \cdot \text{fes}/\text{FES} + \pi/5) \tag{6}$$

where fes is current number of function evaluations. FES is the maximum number of function evaluations. The size of the transformation parameter time changes with the evaluation of the population. From Lines 12−18, the neighborhood size and archive content are dynamically updated by evolutionary experience. And the algorithm updates the position and fitness value of the particles. The algorithm ends with the output of the global optimal value.

3.3 Evolutionary experience-driven adaptive search structure

It is known from Section 3.1 that for many changes in the neighborhood structure, dynamic improvements based on distance and the number of stops are often used. Specifically, the corresponding particles are selected for learning in a population of particles that

Algorithm 1 Proposed EEDSPSO

Input: Control parameters: Ns (swarm size), FES (maximum number of function evaluations), $[x^d_{\min}, x^d_{\max}]$(position boundary), $[v^d_{\min}, v^d_{\max}]$(velocity boundary), Nh(the neighborhood size), Ea(the elite archive), and PB(personal historical best position)

Output: Optimal value

```
Initialize the swarm;
for fes = 1,2,3,...,FES do
    PB_n → the best particle in Nh;
    PB_arch → the random particle in Ne;
    for i = 1; i < Ns; i++ do
        if rand(1) < time (calculate by Eq. (6)) then
            Update the particles velocity using Eq. (8);
        else
            Update the particles velocity using Eqs. (11) and (12);
        end if
    end for
    Update PB and evolutionary experience;
    Select progressive particles to update the neighborhood Nh;
    Sort PB according to the evolutionary experience;
    Update the elite archive Ea;
    Calculate the particles' position using Eq. (2);
end for
return global optimal value;
```

are close to each other relative to themselves. Or the topology of the population is dynamically updated when the population reaches a certain number of stagnation generations. And above dynamic changes usually ignore the degree of progress in each evolutionary process, namely, the evolutionary experience of the population.

An example of an evolutionary experience-driven structure is shown in Fig. 1 for a better understanding. As shown in Fig. 1, there are a total of 4 particles X_1, X_2, X_3, and X_4. The fitness values of each particle are {5, 9, 10, and 15} after the t iteration, which becomes {4, 6, 8, and 11} after $t+1$ iteration. In this paper, the algorithm considers the minimum value optimization problem. The difference in fitness at each iteration is the degree of evolution of the particle. The degree of evolution of the four particles in Fig. 1 after the $t+1$ iteration is {1, 3, 2, 4}. Although X_1's fitness value is better, X_4's evolutionary degree of the particle with the worst fitness value is better. It means X_4 has a better evolutionary experience.

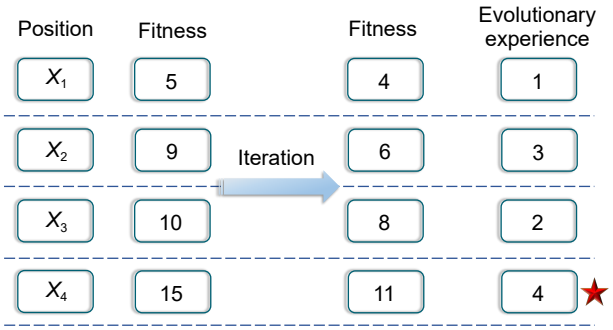


Fig. 1 Evolutionary experience-driven schematic.

3.3.1 Experience-based neighborhood topology adjustment

An important concept, namely evolutionary experience, is introduced in the above paper. The improvement of algorithms using evolutionary experience is a novel attempt. In this section, experience-based neighborhood topology adjustment and Cauchy mutation are introduced in detail.

The neighborhood topology's importance is specifically shown by the number of neighborhood particles that has a serious impact on the algorithm's diversity and convergence. Figure 2 shows the different states of the population after iteration with different numbers of neighborhood particles. Red particles represent the historical optimal positions with better fitness values in that neighborhood range, and the yellow particles represent the particles to be evolved, where the yellow particles learn from the red particles. From Fig. 2a, it can be observed that when the number of particles in the neighborhood is small, a relatively large number of red examples will be generated. Then the population will converge to different regions in the search space, so that the diversity of the population is better maintained. On the contrary, it can be seen from Fig. 2b that the population possesses better convergence when the number of neighboring particles is higher. And the requirements of diversity and convergence of populations are different at different times of evolution. So, it is extremely important to

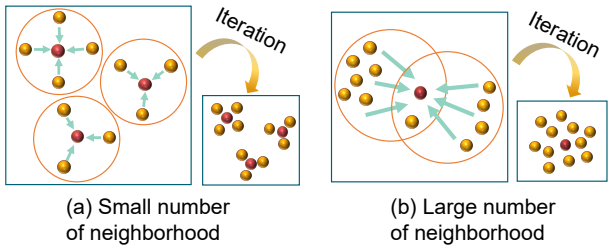


Fig. 2 Small and large number of neighborhoods.

control the range size of the neighborhood reasonably well.

Based on the above analysis, an evolutionary experience-driven framework is used to control the size of the neighborhood range. Firstly, the difference is made between the before and after fitness values of the particles after each iteration. Then the values of the neighborhood range before each iteration are taken for each particle whose difference value is greater than 0 (i.e., the particle whose fitness value becomes better). The squared average of these values (H) is the base value of the neighborhood size for the next population iteration based on evolutionary experience. The squared average is used instead of the arithmetic average mainly to prevent the effect of extreme values. Finally, in order to allow for a more varied selection of particles, the base value H for each particle is treated using a Cauchy distribution that takes a wide distribution of values. In each loop iteration, the neighborhood size Nh of each individual is generated independently according to the Cauchy distribution with position parameter H and scale parameter 1. And the particle PB_n with the best fitness value in the neighborhood range is selected according to the current neighborhood size. The neighborhood size is defined as follows:

$$\mathrm{Nh} = \mathrm{Cauchy}(H, 1) \tag{7}$$

where H is the squared average of the neighborhood sizes of particles whose fitness values have become better in the previous generation. The initial value of H is 5, and then it is updated at the end of each iteration.

3.3.2 Experience-based elite archive mechanism

The necessity of using evolutionary experience to dynamically update the neighborhood range is described above. For this part it is called the individual cognitive part. The other part, the social cognitive part, is also extremely important. It often plays the role of leading the population to convergence. For the traditional social cognitive part, the global optimum is used to guide learning. This tends to make the population converge faster and fall into local optima. From the above introduction to the archive mechanism, it can be seen that an appropriate archive mechanism can collect and store some excellent information in the population and then guide the population to evolve in a better direction. Therefore, an experience-based elite archive mechanism is used. The experience-based evolutionary framework can be described as follows:

$$\begin{aligned} v_i(t) = {} & c_1 \cdot \mathrm{rand}(1, D) \cdot (\mathrm{PB}_n - x_i(t-1)) + \\ & c_2 \cdot \mathrm{rand}(1, D) \cdot (\mathrm{PB}_{\mathrm{arch}} - x_i(t-1)) \end{aligned} \tag{8}$$

$$c_1 = 2 \cdot (1 - \mathrm{fes}/\mathrm{FES}) \tag{9}$$

$$c_2 = 2 - 0.5 \cdot c_1 \tag{10}$$

where PB_n is the historical optimum with the best fitness value selected from the neighborhood based on the experience-based neighborhood topology adjustment. $\mathrm{PB}_{\mathrm{arch}}$ is the guide particle selected based on the experience-based elite archive mechanism. Firstly, the size of the elite archive is determined, here it is set to 15. Next, the particles with better fitness values in the previous generation are sorted according to the evolutionary degree. The top 15 particles are selected and stored in the Ne archive, and a particle is randomly selected from the archive as a guide particle ($\mathrm{PB}_{\mathrm{arch}}$). If no 15 particles became better in the previous generation update, then the top 10 historical best values with better fitness values are selected and deposited in the archive.

From Eqs. (8)−(10), EEDSPSO combines these two learning methods through a dynamic approach. c_1 and c_2 change dynamically with the number of evaluations. In the early evolutionary stage of the algorithm, c_1 is larger and c_2 is smaller. Neighborhood learning dominates, which facilitates exploration in the early stages of the algorithm, c_1 is smaller and c_2 is larger in the late evolutionary stages. The larger proportion of experience-based elite archive mechanism makes the algorithm have stronger local search capability. Thus, EEDSPSO effectively integrates the neighborhood learning strategy and the elite archive mechanism. It can balance algorithm's diversity and convergence well.

3.4 Gaussian crisscross learning strategy

An exemplary algorithm is one that adeptly maintains a harmonious equilibrium between convergence and diversity. Although assimilating knowledge from proficient particles expedites the algorithm's convergence, it often proves arduous to strike a balance with respect to its diversity. Equation (8) discriminates particles possessing commendable fitness values by virtue of an evolutionary experience-driven adaptive framework, thereby exerting an influence on the algorithm's diversity. To overcome this drawback, a Gaussian crisscross learning strategy is introduced into the algorithm. The Gaussian crisscross learning strategy is defined as follows:

$$v_i(t) = \omega \cdot (v_i(t-1) + N(0,1)) + 2 \cdot \text{rand}(1,D) \cdot \left(\text{PB}_{\text{rp}} - x_i(t-1)\right) \tag{11}$$

$$v_i(t) = \omega \cdot v_i(t-1) + 2 \cdot \text{rand}(1,D) \cdot (\text{PB}_{\text{re}} - x_i(t-1)) \tag{12}$$

$$\omega = 1 - \text{fes}/\text{FES} \tag{13}$$

where ω is the inertia weight component, decreasing gradually from 1 to 0. First a random number rand (1) is generated. If it is smaller than ω, then the population is updated using Eq. (11). That is, the iterative operation is performed with high probability using Eq. (11) in the early stage of population iteration. Equation (11) uses a learning mechanism with Gaussian mutation. This is because it is more biased to enhance the diversity of the algorithm in the early stages of the population. Adding Gaussian terms to the original velocity is conducive to enhancing the discreteness in population iterations and thus more conducive to maintaining population diversity. PB_{rp} is a randomly selected particle in the population that is worse than its own historical optimal value. PB_{re} is a randomly selected particle from a population that is better than its own historical optimum. And the iterative operation is performed using Eq. (12) only when the randomly generated number rand (1) is greater than ω. So, this iterative operation is more biased towards the late iteration of the population, which means that it is more biased towards maintaining the convergence performance of the population. At this point, learning from better particles can further improve the convergence of the population in the late iterative stage.

3.5 Computational complexity analysis

In this paper, the proposed algorithm contains three main components, experience-based neighborhood topology adjustment, experience-based elite archive mechanism, and Gaussian crisscross learning strategy. To analyze the computational complexity of each component, for a problem of dimension D, experience-based neighborhood topology adjustment and experience-based elite archive mechanism use an adaptive framework of evolutionary experience. Evolutionary experience adaptive framework requires sorting according to their fitness values. The computational complexity of the sort operation is $O(n\log_2 n)$. In addition, Gaussian crisscross learning strategy is a new learning strategy to further balance the diversity and convergence. Obviously, the computational complexity of this strategy is $O(1)$ in every generation. Therefore, based on the computational complexity analysis of the above main parts, in every generation, the computational complexity of the proposed algorithm is $O(n\log_2 n)$.

4 Experiment

4.1 Experimental settings

This section analyzes the performance of the algorithm through a series of experiments. Experiment 1 analyzes the effectiveness of the algorithm with different strategies to verify the rationality of the proposed strategies. Experiment 2 analyzes the effectiveness of the algorithm's parameters to select more suitable parameter values. Experiment 3 compares different algorithms on the CEC2013 test set to verify the excellent performance of the proposed algorithm. In Experiment 4, in order to verify the superiority of the proposed algorithm in different test problems, CEC2017 test set is used to carry out the comparative experiment of the proposed algorithm. Experiment 5 uses non-parametric tests to compare the significant differences between the proposed algorithm and other algorithms in CEC test sets.

In this experiment, seven popular PSO variants are selected as comparison algorithms. They are used to verify the comprehensive performance of EEDSPSO. The parameter settings of each algorithm are shown in Table 1. And the comparison experiments are done in two widely used test sets, CEC2013 test set and CEC2017 test set. Among them, two sets of experiments with dimension D as 50 are done. To further verify the effectiveness of the algorithm in higher dimensions, experiments with dimension 100 are also done using the CEC2017. In order to comprehensively assess the performance of the proposed algorithm, a series of experiments is conducted utilizing the CEC2013 and CEC2017 test sets across varying dimensions. These test sets are renowned benchmarks specifically designed to evaluate the efficacy of optimization algorithms. By subjecting the algorithm to diverse problem domains and dimensionalities, a rigorous evaluation is undertaken to gauge its adaptability and robustness. This empirical approach allows for a comprehensive examination of the algorithm's capabilities and effectiveness, thereby providing a comprehensive understanding of its performance across different problem landscapes. To ensure the fairness and accuracy of the experiments,

Table 1 Parameter settings for all algorithms.

Algorithm	Parameter setting
EEDSPSO	ω=[1, 0], c_1=2 · ω, and c_2=[1, 2]
BFLPSO[34]	ω=[0.2, 0.9], c=1.494 45, I=E=1, and G=5
HCLDMSPSO[35]	ω=[0.29, 0.99], ω_2, c_1=c_2=[0.5, 0.25], P_m=0.1, and $V_{\max}$=0.5 · Range
DSPSO[36]	c_1=2.0 and $F_{\min}$=0.7
MPSO[37]	logistic chaotic ω=[0.4, 0.9] and c_1=c_2=2.0
BLPSO[38]	ω=[0.2, 0.9], c=1.494 45, and G=5
GLPSO[39]	ω=0.7298, c=1.496 18, pm=0.01, and sg=7
HCLPSO[40]	ω=[0.2, 0.99], c_1=[0.5, 2.5], c_2=[0.5, 2.5], and c=[1.5, 3]

each algorithm is run 51 times independently on each test set. The maximum evaluation test of the algorithm is set to 10 000×D.

4.2 Strategy validity analysis

From the introduction of the algorithms in Section 3, it is clear that the good performance of EEDSPSO is mainly due to the three newly introduced strategies, namely, the experience-based neighborhood topology adjustment, the experience-based elite archive mechanism, and the Gaussian crisscross learning strategy. Therefore, in this section, some experiments are used to examine the performance effects of different strategies on populations. To examine the performance of these strategies in an integrated manner, this experiment selects four different test functions in CEC2017, namely, unimodal function $f(1)$, simple multimodal function $f(8)$, hybrid function $f(13)$, and composition function $f(29)$. The results are shown in Figs. 3 and 4, where GCL, EEA, and ENT are the removal of the Gaussian crisscross learning strategy, the removal of the experience-based elite archive mechanism, and the algorithms for removing the experience-based neighborhood topology adjustment.

In return for analyzing the performance of the three new strategies, two metrics, namely the fitness value and diversity, are used for testing[41].

The results presented in Fig. 3 indicate that ENT and EEA possess poor convergence accuracy relative to EEDSPSO and GCL. These results indicate that experience-based neighborhood topology adjustment and experience-based elite archive mechanism have significant performance improvement on the algorithm performance. EEDSPSO slightly outperforms GCL on the unimodal function. The performance is closer to the other three functions, but EEDSPSO still has the best performance. Thus, the Gaussian crisscross learning strategy further enhances the algorithm's performance

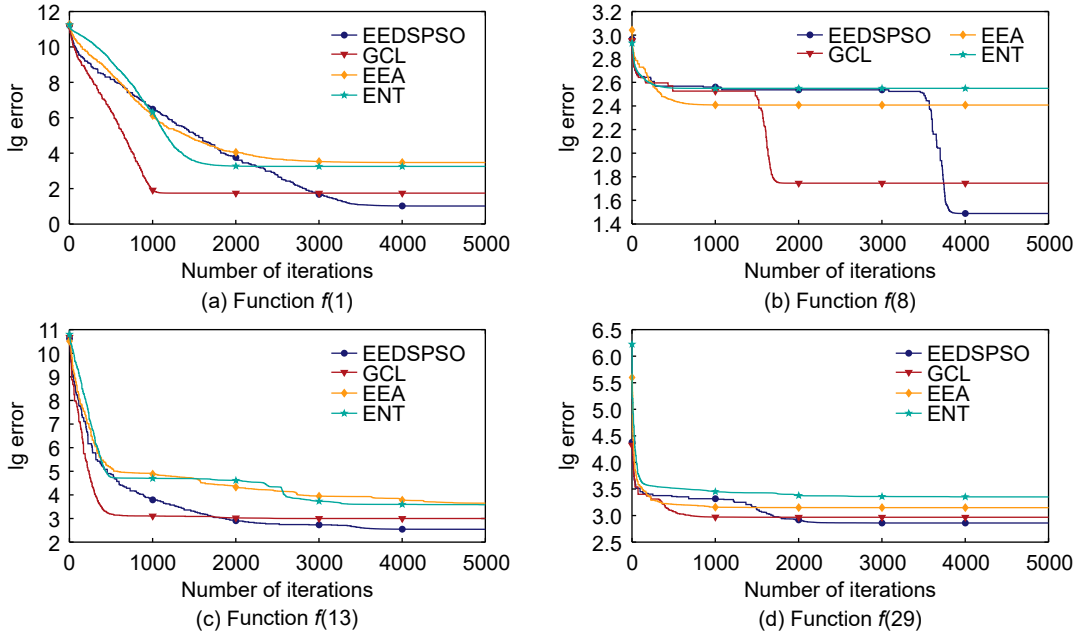


Fig. 3 Optimization values of EEDSPSO, GCL, EEA, and ENT.

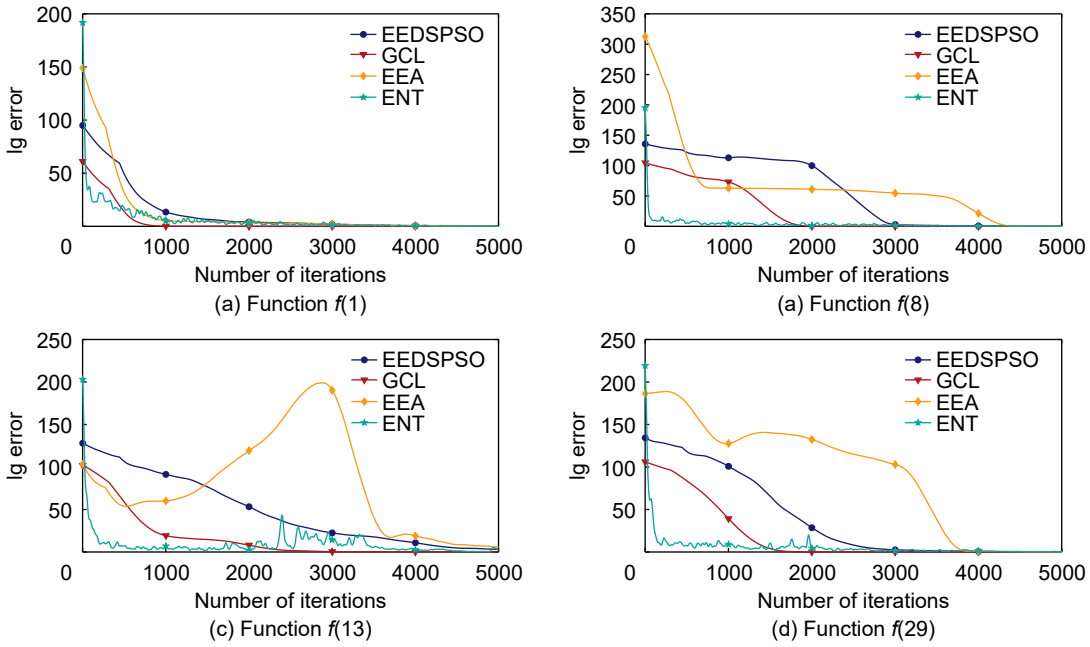


Fig. 4 Distance between the particles of EEDSPSO, GCL, EEA, and ENT.

in an evolutionary experience-driven adaptive framework, in particular, the improvement of the algorithm's convergence accuracy. In addition, the convergence rate of EEDSPSO is relatively smooth compared to the other three algorithms. And EEDSPSO always stalls last in the late stage of the algorithm. A conclusion can be drawn that EEDSPSO plays an excellent role in balancing the algorithm's diversity and convergence.

In order to make the data more intuitive, the smoothdata function is used to smooth the data. The results in Fig. 4 are generally more similar to the performance of the results in Fig. 3, where EEA shows more confusion in diversity. For example, in the hybrid function, EEA has a low diversity in the early stage, but the diversity becomes abnormally large in the later stage around 3000 generations. This is clearly detrimental to the eventual convergence of the population, so it is clearly easy to see in Fig. 3 that EEA eventually possesses the worst convergence accuracy. This also directly indicates that the experience-based elite archive mechanism has a positive effect on the convergence of the population. Similarly, ENT's diversity is also more chaotic and performs the worst in terms of diversity accuracy. This also indicates that experience-based neighborhood topology adjustment provides better diversity for the population. GCL and EEDSPSO are similar to the experiments above, indicating that the Gaussian crisscross learning strategy further improves the performance of the algorithm. EEDSPSO is still the smoothest and best performing algorithm.

The findings pertaining to diversity and convergence substantiate the commendable efficacy of EEDSPSO. In essence, the pioneering topological approach effectively upholds population diversity, thereby ensuring its robustness. Furthermore, the incorporation of the experience-based elite archive mechanism enhances population convergence, fostering increased efficiency. Moreover, the employment of the Gaussian crisscross learning strategy serves to bolster the algorithm's overall performance. The results of diversity and convergence verify that EEDSPSO has very good performance. Overall, the novel topology strategy maintains the diversity of the population. The experience-based elite archive mechanism improves the convergence of the population. The Gaussian crisscross learning strategy further improves the overall performance of the algorithm.

4.3 Parameter validity analysis

In the EEDSPSO algorithm, an adaptive framework based on evolutionary experience-driven and a Gaussian crisscross learning strategy are applied to co-evolve the population, which well balances the diversity and convergence of the population. In this section, two parameters, namely the adaptive parameter time and the archive size Ea, are selected to verify the impact of different parameters on the performance of the algorithm. Comparison experiments are performed

using different test functions in CEC2017, i.e., unimodal function $f(1)$, simple multimodal function $f(9)$, hybrid function $f(17)$, and composition function $f(21)$. The algorithms have the same settings except for the different settings of the comparison parameters.

One of the important parameters is the archive size of the elite archive. The effect of the archive size Ea on this strategy is extremely significant. When the archive size is set larger, the selection range of elite particles becomes larger, and then it is easier to select poorer particles. Thus, the convergence of the algorithm becomes worse. On the contrary, when the archive size becomes smaller, then the algorithm converges faster, and thus it is very easy to fall into local optima. Therefore, this section sets multiple archive size values for comparison experiments. The experimental results are shown in Figs. 5 and 6.

From Fig. 5, it can be seen that four archive sizes of 5, 15, 25, and 35 based on CEC2017 are selected for comparison experiments. For the unimodal function $f(1)$, the performance of the algorithm using the four parameters is basically the same. Among them, the algorithm performs slightly better when Ea is taken as 15 compared to using other other parameters. In the simple multimodal function $f(9)$, the algorithm performs the worst when Ea is taken as 5. This indicates that when the archive size is small, the algorithm is more likely to learn from elite particles. Thus, it converges faster and falls into a local optimum. When Ea is taken as 15, the algorithm performs better in terms of convergence speed and convergence accuracy. For the hybrid function $f(17)$ and the composition function $f(21)$, the algorithm performs the best when Ea is taken as 15, and the worst when Ea is taken as 5. It indicates that as the archive size becomes larger, the performance of the algorithm is limited as well. The main reason is that as the archive size becomes larger, the algorithm is unable to select the appropriate elite particles for learning, which leads to a decrease in the performance of the algorithm. When Ea is taken as 5, the algorithm still has the problem of faster convergence and lower accuracy. A smaller archive capacity is not conducive to the performance of this algorithm. On the contrary, when Ea is 15, the algorithm has excellent convergence speed and the best convergence accuracy under different types of functions. Therefore, the archive size is finally set to 15 for the elite archive.

The time is an adaptive parameter, which is used to reasonably control the weight of the evolutionary experience adaptive framework and the Gaussian crisscross learning strategy in the population iterations. The parameter time is controlled using Eq. (6), and the value of time is large in the early stage and becomes less volatile with population iteration. This means that, in the early stage of the population, the algorithm mainly uses an adaptive framework driven by evolutionary experience. Later in the population, the weight of the Gaussian crisscross learning strategy slowly becomes larger. For the effectiveness of this

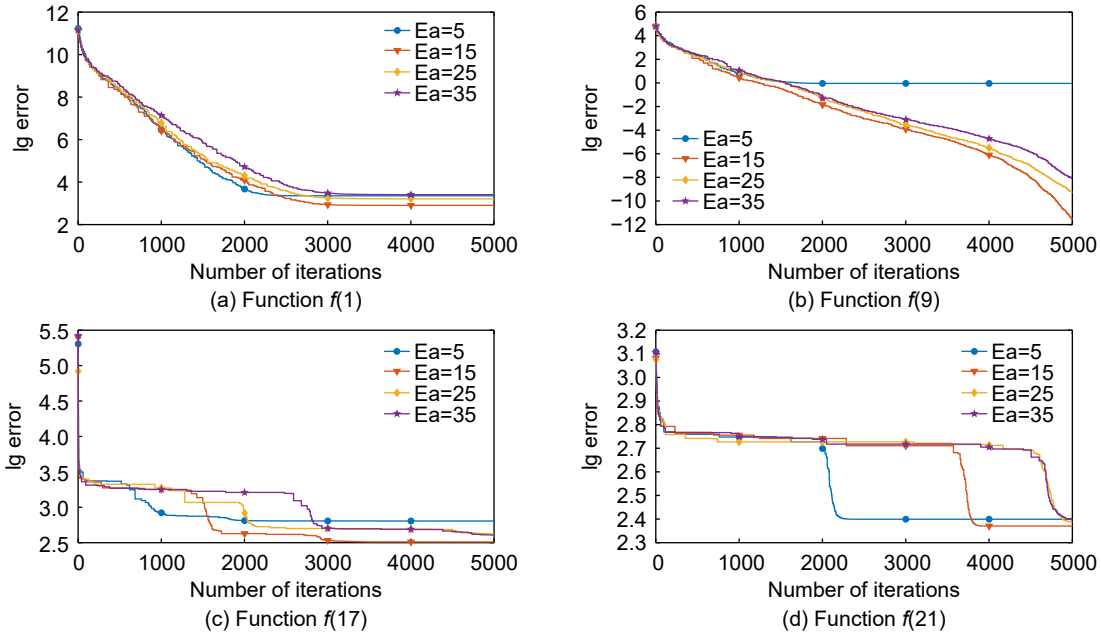


Fig. 5 Optimization values under different archive sizes.

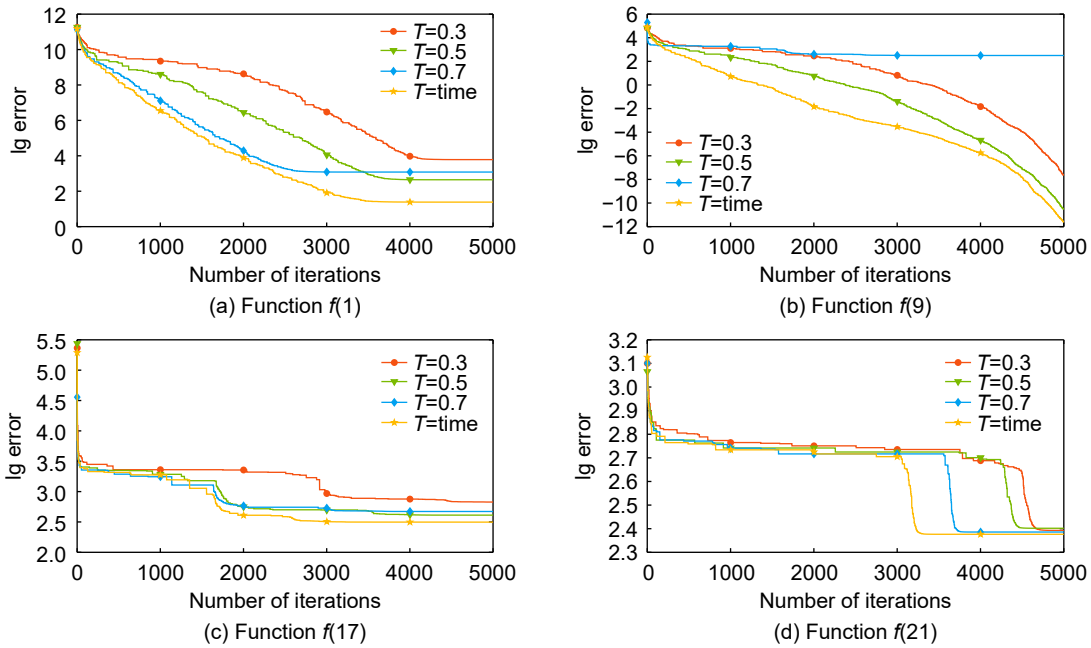


Fig. 6 Optimization values under different transition probabilities.

control approach, this section is experimentally verified by three fixed parameters, and the experimental results are shown in Fig. 6.

From Fig. 6, three fixed parameters are set for comparison experiments, 0.3, 0.5, and 0.7. Firstly, for the unimodal function $f(1)$, the algorithm is the least effective when the transition probability T is taken as 0.3. This indicates that the adaptive framework based on evolutionary experience has less weight when the transition probability is small. The convergence and diversity of the algorithm also become worse. In contrast, the algorithm works the best when the adaptive parameter time is used. Secondly, for the simple multimodal function $f(9)$, the algorithm works worst when T takes 0.7. This is because when T becomes larger, the weight of the Gaussian crisscross learning strategy decreases and thus falls into the local optimal value prematurely. Thus, the diversity and convergence of the algorithm cannot be improved in time. Finally, for the hybrid function $f(17)$ and the composition function $f(21)$, the overall difference is not particularly large. But the algorithm works the best when the transformation probability uses the adaptive parameter time. Especially, the improvement for algorithm diversity and convergence is far better than other parameters. Therefore, this paper uses the adaptive parameter time to dynamically update the weight of the adaptive framework and the Gaussian crisscross learning strategy in the population iteration.

4.4 Comparison of PSO algorithm variants

4.4.1 Data analysis on CEC2013

In purpose of verifying the excellent overall performance of EEDSPSO, this section conducts comparison experiments in the CEC2013 test set and with dimension 50. The mean (Mean) of each independent run of 51 times is used to represent the mean results. For each test function, the results of each algorithm are ranked and presented in Rank. The bolded data are the best ones. Count represents the total number of the best performing test functions for each algorithm. Average rank is the average rank. Total rank stands for the final rank of each algorithm. The final results are shown in Table 2.

From the results presented in Table 2, the following results can be derived. Firstly, for the unimodal functions ($f(1)$–$f(5)$), EEDSPSO achieves the best performance, followed by DSPSO and BLPSO. EEDSPSO achieves a more stable and excellent performance for the functions $f(2)$ and $f(5)$. For the basic multimodal functions ($f(6)$– $f(20)$), the EEDSPSO achieves the best performance on only three tested functions, but the sum of the average rankings is the same as DSPSO. This indicates that EEDSPSO shows excellent performance in solving multimodal functions, but there is still room for improvement. Finally, for more complex composition functions ($f(21)$– $f(28)$), EEDSPSO has the best performance on half of the functions. It directly shows that EEDSPSO is far

Table 2 Experimental results in CEC2013 (D=50).

Function	Metric	HCLPSO	GLPSO	BLPSO	MPSO	DSPSO	HCLDMSPSO	BFLPSO	EEDSPSO
$f(1)$	Mean	5.23×10^{-13}	7.88×10^{-13}	**2.20×10^{-13}**	1.59	3.03×10^{-13}	4.32×10^{-13}	2.27×10^{-13}	2.27×10^{-13}
	Rank	6	7	**1**	8	4	5	2	2
$f(2)$	Mean	1.40×10^{6}	2.16×10^{6}	1.84×10^{7}	2.41×10^{7}	2.37×10^{6}	3.24×10^{6}	1.43×10^{7}	**6.56×10^{5}**
	Rank	2	3	7	8	4	5	6	**1**
$f(3)$	Mean	2.60×10^{8}	2.97×10^{8}	2.90×10^{7}	4.29×10^{8}	**7.97×10^{6}**	1.88×10^{7}	3.44×10^{7}	2.92×10^{7}
	Rank	6	7	3	8	**1**	2	5	4
$f(4)$	Mean	1.57×10^{3}	6.54×10^{3}	2.53×10^{3}	5.33×10^{2}	**5.13×10^{1}**	1.16×10^{4}	2.31×10^{3}	2.19×10^{3}
	Rank	3	7	6	2	**1**	8	5	4
$f(5)$	Mean	7.31×10^{-13}	1.52×10^{-12}	3.15×10^{-13}	1.09×10^{1}	4.09×10^{-13}	4.45×10^{-11}	4.21×10^{-13}	**1.71×10^{-13}**
	Rank	5	6	2	8	3	7	4	**1**
$f(6)$	Mean	4.43×10^{1}	6.31×10^{1}	4.47×10^{1}	9.86×10^{1}	**4.37×10^{1}**	4.53×10^{1}	4.60×10^{1}	4.44×10^{1}
	Rank	2	7	4	8	**1**	5	6	3
$f(7)$	Mean	4.75×10^{1}	6.10×10^{1}	2.05×10^{1}	7.95×10^{1}	1.84×10^{1}	1.08×10^{1}	2.49×10^{1}	**9.80**
	Rank	6	7	4	8	3	2	5	**1**
$f(8)$	Mean	2.11×10^{1}	2.11×10^{1}	**2.05×10^{1}**	2.11×10^{1}	2.11×10^{1}	2.11×10^{1}	2.11×10^{1}	2.10×10^{1}
	Rank	7	6	**1**	3	4	5	8	2
$f(9)$	Mean	4.00×10^{1}	4.02×10^{1}	5.11×10^{1}	5.10×10^{1}	**1.94×10^{1}**	2.72×10^{1}	5.13×10^{1}	1.94×10^{1}
	Rank	4	5	7	6	**1**	3	8	2
$f(10)$	Mean	2.24×10^{-1}	2.41×10^{-1}	2.90×10^{-1}	6.62×10^{-1}	**1.16×10^{-1}**	1.55×10^{-1}	2.56×10^{-1}	1.57×10^{-1}
	Rank	4	5	7	8	**1**	2	6	3
$f(11)$	Mean	2.75	**5.31×10^{-1}**	3.11	6.66×10^{1}	3.76×10^{1}	6.91×10^{1}	1.06	3.49×10^{1}
	Rank	3	**1**	4	7	6	8	2	5
$f(12)$	Mean	1.13×10^{2}	1.44×10^{2}	6.83×10^{1}	1.97×10^{2}	4.60×10^{1}	6.37×10^{1}	5.65×10^{1}	**4.40×10^{1}**
	Rank	6	7	5	8	2	4	3	**1**
$f(13)$	Mean	2.48×10^{2}	2.93×10^{2}	1.33×10^{2}	3.39×10^{2}	1.16×10^{2}	1.50×10^{2}	**1.15×10^{2}**	1.19×10^{2}
	Rank	6	7	4	8	2	5	**1**	3
$f(14)$	Mean	1.07×10^{2}	**1.41×10^{1}**	2.51×10^{2}	2.94×10^{3}	2.81×10^{3}	3.56×10^{3}	2.61×10^{2}	1.37×10^{3}
	Rank	2	**1**	3	7	6	8	4	5
$f(15)$	Mean	7.40×10^{3}	7.60×10^{3}	7.51×10^{3}	7.74×10^{3}	5.50×10^{3}	6.38×10^{3}	7.14×10^{3}	**5.19×10^{3}**
	Rank	5	7	6	8	2	3	4	**1**
$f(16)$	Mean	2.15	1.13	2.08	2.67	3.32	**1.08**	1.93	3.31
	Rank	5	2	4	6	8	**1**	3	7
$f(17)$	Mean	5.40×10^{1}	5.71×10^{1}	**5.00×10^{1}**	1.40×10^{2}	8.19×10^{1}	1.32×10^{2}	5.19×10^{1}	7.22×10^{1}
	Rank	3	4	**1**	8	6	7	2	5
$f(18)$	Mean	1.50×10^{2}	1.58×10^{2}	1.80×10^{2}	2.18×10^{2}	**1.13×10^{2}**	1.49×10^{2}	1.74×10^{2}	3.73×10^{2}
	Rank	3	4	6	7	**1**	2	5	8
$f(19)$	Mean	**2.21**	3.83	3.02	9.08	5.66	6.33	3.81	4.90
	Rank	**1**	4	2	8	6	7	3	5
$f(20)$	Mean	2.03×10^{1}	1.98×10^{1}	**1.85×10^{1}**	2.29×10^{1}	2.76×10^{1}	1.89×10^{1}	1.87×10^{1}	2.04×10^{1}
	Rank	5	4	**1**	7	8	3	2	6
$f(21)$	Mean	6.52×10^{2}	9.56×10^{2}	9.39×10^{2}	2.06×10^{3}	9.70×10^{2}	**6.27×10^{2}**	9.00×10^{2}	9.44×10^{2}
	Rank	2	6	4	8	7	**1**	3	5
$f(22)$	Mean	1.16×10^{2}	**5.05×10^{1}**	2.87×10^{2}	3.13×10^{3}	2.77×10^{3}	3.98×10^{3}	2.55×10^{2}	1.42×10^{3}
	Rank	2	**1**	4	7	6	8	3	5
$f(23)$	Mean	8.82×10^{3}	8.09×10^{3}	7.58×10^{3}	8.96×10^{3}	6.02×10^{3}	6.89×10^{3}	7.45×10^{3}	**4.82×10^{3}**
	Rank	7	6	5	8	2	3	4	**1**

(to be continued)

Table 2 Experimental results in CEC2013 (D=50).

(continued)

Function	Metric	HCLPSO	GLPSO	BLPSO	MPSO	DSPSO	HCLDMSPSO	BFLPSO	EEDSPSO
$f(24)$	Mean	2.81×10^2	3.01×10^2	2.51×10^2	3.24×10^2	2.24×10^2	2.28×10^2	2.52×10^2	**2.22×10^2**
	Rank	6	7	4	8	2	3	5	**1**
$f(25)$	Mean	3.58×10^2	3.38×10^2	3.38×10^2	3.73×10^2	3.64×10^2	3.22×10^2	3.18×10^2	**3.05×10^2**
	Rank	6	5	4	8	7	3	2	**1**
$f(26)$	Mean	**2.00×10^2**	3.31×10^2	2.38×10^2	3.54×10^2	3.39×10^2	3.29×10^2	2.11×10^2	2.69×10^2
	Rank	**1**	6	3	8	7	5	2	4
$f(27)$	Mean	1.22×10^3	1.31×10^3	1.28×10^3	1.48×10^3	8.56×10^2	8.51×10^2	1.04×10^3	**7.35×10^2**
	Rank	5	7	6	8	3	2	4	**1**
$f(28)$	Mean	4.00×10^2	1.29×10^3	**3.91×10^2**	9.92×10^2	6.96×10^2	4.00×10^2	4.00×10^2	4.00×10^2
	Rank	4	8	**1**	7	6	5	2	2
Count		2	3	5	0	6	2	1	**9**
Average rank		4.18	5.25	3.89	7.25	3.93	4.36	3.89	**3.18**
Total rank		5	7	2	8	4	6	2	**1**

superior to other algorithms in handling complex problems. In addition, the performance of DSPSO, which performs consistently in front, fails to make it into the top three and lags far behind other algorithms. All in all, EEDSPSO ranks the first overall and performs well in more complex functions under the experiments with dimension 50 of the CEC2013 test set. This indicates that using an adaptive framework based on evolutionary experience and a Gaussian crisscross learning strategy has excellent performance in solving complex problems.

4.4.2 Data analysis on CEC2017

To further test the performance of the EEDSPSO algorithm, this section performs another experimental analysis using the CEC2017 test set. The experimental method is the same as in the previous section. The experimental results are shown in Tables 3 and 4.

Table 3 provides the experimental results in CEC2017 with dimension 50. First of all, for unimodal functions ($f(1)$ and $f(3)$) and simple multimodal functions ($f(4)$−$f(10)$), EEDSPSO performs better than the other algorithms, as it achieves the best convergence results for 4 of the 9 functions. In contrast, the second ranked DSPSO achieves the best convergence results for only 2 functions. Then for the hybrid function ($f(11)$−$f(20)$), EEDSPSO achieves the best accuracy in 4 out of 10 functions, while the second-ranked HCLPSO performs well in only 3 functions. And DSPSO, which performs reasonably well in simple multimodal functions, does not perform well in one function. Finally, for the more complex composition functions ($f(21)$−$f(30)$), EEDSPSO still outperformed the other algorithms by 4 functions. This is followed by MPSO and HCLDMSPSO both achieving optimal values on 2 functions. In conclusion, EEDSPSO performs much better than the other algorithms for both simple multimodal functions and more complex hybrid and composition functions.

Experiments with both CEC2013 and CEC2017 at 50 dimensions are done above, and the experimental results show that EEDSPSO with the three new strategies outperforms the more popular seven algorithms. To further verify the performance of EEDSPSO in higher dimensions, experiments with CEC2017 at 100 dimensions are conducted. The following information can be summarized from the results in Table 4. Firstly, for unimodal functions ($f(1)$ and $f(3)$) and simple multimodal functions ($f(4)$−$f(10)$), EEDSPSO achieves the highest accuracy on 4 functions, which is better than other algorithms. And only for the function $f(3)$, EEDSPSO ranks the 5th, while the remaining functions are in the top three. This directly indicates that EEDSPSO is extremely good in both the performance of individual functions and overall performance. Then for the hybrid functions ($f(11)$−$f(20)$), EEDSPSO dominates on 4 functions. Both HCLPSO and BFLPSO are tied for second place with excellent performance on 2 functions. Finally, there are the composition functions ($f(21)$−$f(30)$), where EEDSPSO performs the best on the functions $f(25)$−$f(28)$. In addition, the average ranking of EEDSPSO is 2.39, which is also much better than the 3.18 of CEC2013 in 50 dimensions and 3.25 of CEC2017 in 50 dimensions. It is obvious that the performance of EEDSPSO is rather better as the dimensionality increases.

Table 3 Experimental results in CEC2017 (D=50).

Function	Metric	HCLPSO	GLPSO	BLPSO	MPSO	DSPSO	HCLDMSPSO	BFLPSO	EEDSPSO
$f(1)$	Mean	4.79×10^{3}	1.33×10^{3}	1.71×10^{3}	2.62×10^{9}	2.34×10^{3}	1.79×10^{3}	**1.32×10^{3}**	1.70×10^{3}
	Rank	7	2	4	8	6	5	**1**	3
$f(3)$	Mean	**3.87**	7.63×10^{2}	3.13×10^{4}	8.24×10^{1}	1.42×10^{1}	1.63×10^{4}	3.80×10^{4}	1.35×10^{4}
	Rank	**1**	4	7	3	2	6	8	5
$f(4)$	Mean	9.76×10^{1}	9.09×10^{1}	1.17×10^{2}	5.34×10^{2}	2.28×10^{2}	1.04×10^{2}	1.24×10^{2}	**8.80×10^{1}**
	Rank	3	2	5	8	7	4	6	**1**
$f(5)$	Mean	9.13×10^{1}	1.05×10^{2}	7.13×10^{1}	1.47×10^{2}	**3.38×10^{1}**	6.40×10^{1}	6.80×10^{1}	6.41×10^{1}
	Rank	6	7	5	8	**1**	2	4	3
$f(6)$	Mean	1.27×10^{-5}	2.20×10^{-3}	**1.01×10^{-8}**	1.49	2.37×10^{-3}	2.89×10^{-2}	2.72×10^{-8}	4.18×10^{-3}
	Rank	3	4	**1**	8	5	7	2	6
$f(7)$	Mean	1.53×10^{2}	1.28×10^{2}	1.32×10^{2}	2.47×10^{2}	**7.05×10^{1}**	1.06×10^{2}	1.31×10^{2}	2.02×10^{2}
	Rank	6	3	5	8	**1**	2	4	7
$f(8)$	Mean	8.41×10^{1}	9.97×10^{1}	7.32×10^{1}	1.67×10^{2}	3.34×10^{1}	6.59×10^{1}	7.16×10^{1}	**3.23×10^{1}**
	Rank	6	7	5	8	2	3	4	**1**
$f(9)$	Mean	2.12×10^{1}	1.70×10^{2}	6.42×10^{-1}	9.46×10^{2}	8.50×10^{-1}	4.38×10^{-1}	5.40×10^{-1}	**9.02×10^{-2}**
	Rank	6	7	4	8	5	2	3	**1**
$f(10)$	Mean	4.65×10^{3}	4.22×10^{3}	5.00×10^{3}	5.51×10^{3}	3.33×10^{3}	4.73×10^{3}	4.67×10^{3}	**2.47×10^{3}**
	Rank	4	3	7	8	2	6	5	**1**
$f(11)$	Mean	1.19×10^{2}	1.02×10^{2}	6.70×10^{1}	2.93×10^{2}	1.43×10^{2}	1.05×10^{2}	5.44×10^{1}	**5.36×10^{1}**
	Rank	6	4	3	8	7	5	2	**1**
$f(12)$	Mean	**2.83×10^{5}**	2.95×10^{5}	1.53×10^{6}	3.43×10^{7}	3.55×10^{5}	8.83×10^{5}	1.51×10^{6}	1.17×10^{6}
	Rank	**1**	2	7	8	3	4	6	5
$f(13)$	Mean	2.49×10^{3}	4.26×10^{3}	**8.65×10^{2}**	2.20×10^{4}	2.69×10^{3}	1.90×10^{3}	1.01×10^{3}	1.22×10^{3}
	Rank	5	7	**1**	8	6	4	2	3
$f(14)$	Mean	2.19×10^{4}	1.37×10^{4}	9.05×10^{4}	**9.77×10^{2}**	1.05×10^{4}	3.41×10^{4}	8.82×10^{4}	2.23×10^{4}
	Rank	4	3	8	**1**	2	6	7	5
$f(15)$	Mean	**1.31×10^{3}**	5.65×10^{3}	5.53×10^{3}	3.50×10^{3}	2.60×10^{3}	3.01×10^{3}	4.04×10^{3}	3.15×10^{3}
	Rank	**1**	8	7	5	2	3	6	4
$f(16)$	Mean	1.00×10^{3}	9.97×10^{2}	7.24×10^{2}	1.38×10^{3}	5.10×10^{2}	7.34×10^{2}	7.53×10^{2}	**5.06×10^{2}**
	Rank	7	6	3	8	2	4	5	**1**
$f(17)$	Mean	8.19×10^{2}	7.65×10^{2}	4.50×10^{2}	9.15×10^{2}	5.15×10^{2}	6.07×10^{2}	4.95×10^{2}	**3.83×10^{2}**
	Rank	7	6	2	8	4	5	3	**1**
$f(18)$	Mean	5.95×10^{4}	6.57×10^{4}	6.50×10^{5}	**4.88×10^{4}**	6.51×10^{4}	4.44×10^{5}	3.38×10^{5}	7.97×10^{5}
	Rank	2	4	7	**1**	3	6	5	8
$f(19)$	Mean	**1.47×10^{3}**	1.50×10^{4}	1.65×10^{4}	4.75×10^{3}	1.39×10^{4}	8.50×10^{3}	1.03×10^{4}	1.49×10^{4}
	Rank	**1**	7	8	2	5	3	4	6
$f(20)$	Mean	5.25×10^{2}	5.06×10^{2}	2.76×10^{2}	8.15×10^{2}	2.12×10^{2}	4.35×10^{2}	2.71×10^{2}	**1.53×10^{2}**
	Rank	7	6	4	8	2	5	3	**1**
$f(21)$	Mean	2.87×10^{2}	2.75×10^{2}	2.67×10^{2}	5.67×10^{2}	**2.34×10^{2}**	2.68×10^{2}	2.73×10^{2}	2.68×10^{2}
	Rank	7	6	2	8	**1**	3	5	4
$f(22)$	Mean	3.99×10^{3}	3.40×10^{3}	4.93×10^{3}	1.80×10^{2}	6.80×10^{2}	3.18×10^{3}	4.77×10^{3}	**1.00×10^{2}**
	Rank	6	5	8	2	3	4	7	**1**
$f(23)$	Mean	5.27×10^{2}	5.19×10^{2}	4.89×10^{2}	1.07×10^{3}	4.69×10^{2}	5.03×10^{2}	4.97×10^{2}	**4.57×10^{2}**
	Rank	7	6	3	8	2	5	4	**1**
$f(24)$	Mean	5.97×10^{2}	5.99×10^{2}	5.61×10^{2}	**2.71×10^{2}**	5.25×10^{2}	5.71×10^{2}	5.66×10^{2}	5.30×10^{2}
	Rank	7	8	4	**1**	2	6	5	3

(to be continued)

Table 3 Experimental results in CEC2017 (D=50).

(continued)

Function	Metric	HCLPSO	GLPSO	BLPSO	MPSO	DSPSO	HCLDMSPSO	BFLPSO	EEDSPSO
$f(25)$	Mean	5.18×10^{2}	5.68×10^{2}	5.34×10^{2}	7.01×10^{2}	5.84×10^{2}	**5.13×10^{2}**	5.45×10^{2}	5.57×10^{2}
	Rank	2	6	3	8	7	**1**	4	5
$f(26)$	Mean	2.01×10^{3}	1.42×10^{3}	1.73×10^{3}	5.93×10^{2}	5.99×10^{2}	1.70×10^{3}	1.73×10^{3}	**4.14×10^{2}**
	Rank	8	4	6	2	3	5	7	**1**
$f(27)$	Mean	6.17×10^{2}	6.68×10^{2}	5.59×10^{2}	1.67×10^{3}	8.07×10^{2}	6.23×10^{2}	5.72×10^{2}	**5.42×10^{2}**
	Rank	4	6	2	8	7	5	3	**1**
$f(28)$	Mean	4.88×10^{2}	5.06×10^{2}	5.04×10^{2}	1.62×10^{3}	5.48×10^{2}	**4.81×10^{2}**	5.14×10^{2}	5.00×10^{2}
	Rank	2	5	4	8	7	**1**	6	3
$f(29)$	Mean	6.71×10^{2}	7.71×10^{2}	**4.42×10^{2}**	1.20×10^{3}	6.31×10^{2}	6.70×10^{2}	4.57×10^{2}	6.00×10^{2}
	Rank	6	7	**1**	8	4	5	2	3
$f(30)$	Mean	7.83×10^{5}	8.04×10^{5}	8.11×10^{5}	**2.88×10^{5}**	4.06×10^{6}	8.37×10^{5}	7.84×10^{5}	8.27×10^{5}
	Rank	2	4	5	**1**	8	7	3	6
Count		4	0	3	4	3	2	1	**12**
Average rank		4.79	5.32	4.68	6.36	3.96	4.43	4.50	**3.25**
Total rank		6	7	5	8	2	3	4	**1**

Table 4 Experimental results in CEC2017 (D=100).

Function	Metric	HCLPSO	GLPSO	BLPSO	MPSO	DSPSO	HCLDMSPSO	BFLPSO	EEDSPSO
$f(1)$	Mean	6.99×10^{3}	5.71×10^{3}	4.00×10^{3}	2.04×10^{10}	7.37×10^{3}	6.63×10^{3}	4.31×10^{3}	**3.78×10^{3}**
	Rank	6	4	2	8	7	5	3	**1**
$f(3)$	Mean	**1.71×10^{3}**	8.57×10^{4}	1.68×10^{5}	2.01×10^{4}	6.43×10^{3}	1.84×10^{5}	1.92×10^{5}	9.88×10^{4}
	Rank	**1**	4	6	3	2	7	8	5
$f(4)$	Mean	6.27×10^{2}	2.63×10^{2}	2.41×10^{2}	2.57×10^{3}	4.93×10^{2}	6.46×10^{2}	2.45×10^{2}	**2.41×10^{2}**
	Rank	6	4	2	8	5	7	3	**1**
$f(5)$	Mean	7.45×10^{2}	3.01×10^{2}	1.98×10^{2}	4.42×10^{2}	**8.65×10^{1}**	7.12×10^{2}	1.84×10^{2}	1.14×10^{2}
	Rank	8	5	4	6	**1**	7	3	2
$f(6)$	Mean	6.00×10^{2}	1.37×10^{-1}	1.44×10^{-6}	1.31×10^{1}	1.60×10^{-2}	6.01×10^{2}	**7.97×10^{-7}**	3.30×10^{-3}
	Rank	7	5	2	6	4	8	**1**	3
$f(7)$	Mean	1.07×10^{3}	4.26×10^{2}	3.06×10^{2}	8.86×10^{2}	**1.60×10^{2}**	9.61×10^{2}	2.97×10^{2}	2.86×10^{2}
	Rank	8	5	4	6	**1**	7	3	2
$f(8)$	Mean	1.05×10^{3}	2.92×10^{2}	1.95×10^{2}	4.94×10^{2}	**9.05×10^{1}**	1.02×10^{3}	1.81×10^{2}	1.12×10^{2}
	Rank	8	5	4	6	**1**	7	3	2
$f(9)$	Mean	1.44×10^{3}	4.48×10^{3}	1.15×10^{1}	6.04×10^{3}	1.04×10^{1}	1.26×10^{3}	1.64×10^{1}	**4.59×10^{0}**
	Rank	6	7	3	8	2	5	4	**1**
$f(10)$	Mean	1.29×10^{4}	1.10×10^{4}	1.27×10^{4}	1.35×10^{4}	8.23×10^{3}	1.30×10^{4}	1.23×10^{4}	**6.58×10^{3}**
	Rank	6	3	5	8	2	7	4	**1**
$f(11)$	Mean	1.94×10^{3}	4.06×10^{2}	5.78×10^{2}	1.49×10^{3}	1.04×10^{3}	2.07×10^{3}	4.87×10^{2}	**3.29×10^{2}**
	Rank	7	2	4	6	5	8	3	**1**
$f(12)$	Mean	**5.35×10^{5}**	2.24×10^{6}	4.13×10^{6}	1.05×10^{9}	3.23×10^{6}	1.94×10^{6}	3.95×10^{6}	2.31×10^{6}
	Rank	**1**	3	7	8	5	2	6	4
$f(13)$	Mean	4.95×10^{3}	3.34×10^{3}	2.51×10^{3}	2.06×10^{7}	3.96×10^{3}	5.60×10^{3}	**2.48×10^{3}**	3.12×10^{3}
	Rank	6	4	2	8	5	7	**1**	3
$f(14)$	Mean	8.06×10^{4}	1.54×10^{5}	1.96×10^{6}	1.56×10^{5}	**7.10×10^{4}**	3.54×10^{5}	1.61×10^{6}	1.78×10^{5}
	Rank	2	3	8	4	**1**	6	7	5
$f(15)$	Mean	2.66×10^{3}	1.66×10^{3}	7.26×10^{2}	3.85×10^{4}	2.20×10^{3}	3.07×10^{3}	**6.54×10^{2}**	1.19×10^{3}
	Rank	6	4	2	8	5	7	**1**	3

(to be continued)

Table 4 Experimental results in CEC2017 (D=100).

(continued)

Function	Metric	HCLPSO	GLPSO	BLPSO	MPSO	DSPSO	HCLDMSPSO	BFLPSO	EEDSPSO
$f(16)$	Mean	4.58×10^{3}	2.72×10^{3}	2.40×10^{3}	3.96×10^{3}	1.54×10^{3}	4.27×10^{3}	2.44×10^{3}	**1.53×10^{3}**
	Rank	8	5	3	6	2	7	4	**1**
$f(17)$	Mean	4.00×10^{3}	2.11×10^{3}	1.59×10^{3}	3.21×10^{3}	1.41×10^{3}	3.78×10^{3}	1.59×10^{3}	**1.14×10^{3}**
	Rank	8	5	3	6	2	7	4	**1**
$f(18)$	Mean	**1.62×10^{5}**	6.56×10^{5}	1.99×10^{6}	2.88×10^{5}	1.96×10^{5}	1.05×10^{6}	1.23×10^{6}	1.22×10^{6}
	Rank	**1**	4	8	3	2	5	7	6
$f(19)$	Mean	2.44×10^{3}	2.81×10^{3}	**9.07×10^{2}**	2.93×10^{5}	1.54×10^{3}	3.09×10^{3}	1.05×10^{3}	1.24×10^{3}
	Rank	5	6	**1**	8	4	7	2	3
$f(20)$	Mean	4.24×10^{3}	2.11×10^{3}	1.66×10^{3}	2.11×10^{3}	9.56×10^{2}	4.08×10^{3}	1.69×10^{3}	**8.28×10^{2}**
	Rank	8	6	3	5	2	7	4	**1**
$f(21)$	Mean	2.58×10^{3}	4.71×10^{2}	4.10×10^{2}	**1.49×10^{2}**	3.20×10^{2}	2.56×10^{3}	4.18×10^{2}	3.35×10^{2}
	Rank	8	6	4	**1**	2	7	5	3
$f(22)$	Mean	1.60×10^{4}	1.13×10^{4}	1.39×10^{4}	**1.49×10^{2}**	2.26×10^{3}	1.27×10^{4}	1.34×10^{4}	1.11×10^{3}
	Rank	8	4	7	**1**	3	5	6	2
$f(23)$	Mean	3.08×10^{3}	7.34×10^{2}	**6.43×10^{2}**	3.02×10^{3}	6.91×10^{2}	9.08×10^{2}	6.53×10^{2}	6.43×10^{2}
	Rank	8	5	**1**	7	4	6	3	2
$f(24)$	Mean	3.60×10^{3}	1.17×10^{3}	1.05×10^{3}	**5.38×10^{2}**	9.60×10^{2}	1.17×10^{3}	1.07×10^{3}	9.33×10^{2}
	Rank	8	7	4	**1**	3	6	5	2
$f(25)$	Mean	3.27×10^{3}	8.19×10^{2}	7.99×10^{2}	3.66×10^{3}	1.07×10^{3}	7.68×10^{2}	8.15×10^{2}	**7.48×10^{2}**
	Rank	7	5	3	8	6	2	4	**1**
$f(26)$	Mean	8.79×10^{3}	9.23×10^{3}	4.95×10^{3}	4.77×10^{3}	2.54×10^{3}	5.67×10^{3}	4.91×10^{3}	**1.32×10^{3}**
	Rank	7	8	5	3	2	6	4	**1**
$f(27)$	Mean	3.50×10^{3}	8.64×10^{2}	6.67×10^{2}	3.27×10^{3}	9.40×10^{2}	7.89×10^{2}	6.84×10^{2}	**6.45×10^{2}**
	Rank	8	5	2	7	6	4	3	**1**
$f(28)$	Mean	3.37×10^{3}	6.09×10^{2}	6.18×10^{2}	4.13×10^{3}	8.57×10^{2}	5.87×10^{2}	6.38×10^{2}	**5.71×10^{2}**
	Rank	7	3	4	8	6	2	5	**1**
$f(29)$	Mean	2.86×10^{3}	2.62×10^{3}	**1.82×10^{3}**	2.71×10^{3}	2.07×10^{3}	2.71×10^{3}	1.82×10^{3}	1.83×10^{3}
	Rank	8	5	**1**	7	4	6	2	3
$f(30)$	Mean	**5.55×10^{3}**	6.15×10^{3}	7.17×10^{3}	1.58×10^{6}	3.28×10^{4}	6.31×10^{3}	7.60×10^{3}	7.45×10^{3}
	Rank	**1**	2	4	8	7	3	6	5
Count		4	0	3	3	4	0	3	**12**
Average rank		6.36	4.79	3.86	6.14	3.61	6.07	4.07	**2.39**
Total rank		8	5	3	7	2	6	4	**1**

4.5 Non-parametric test

To compare the significant differences between EEDSPSO and other algorithms, the results of Tables 2−4 are processed using the Wilcoxon sign test. Tables 5−7 show the corresponding results. R^{+} and R^{-} represent the positive and negative rankings, respectively. n, w, t, and l represent the overall number of functions, the number of functions that perform well with EEDSPSO, the number of functions that perform fairly well, and the number of functions that perform poorly with EEDSPSO, respectively. In addition, the significance level is taken as 0.05.

From Table 5, the significant differences between EEDSPSO and GLPSO, MPSO, DSPSO, and HCLDMSPSO are 2.42×10^{-2}, 2.69×10^{-4}, 1.98×10^{-2}, and 1.33×10^{-2}, respectively, which are less than 0.05. And the number of functions in which EEDSPSO outperforms them is much higher. This directly indicates that EEDSPSO is significantly better than GLPSO, MPSO, DSPSO, and HCLDMSPSO. The significant differences between EEDSPSO and HCLPSO, BLPSO, and BFLPSO are 5.40×10^{-1}, 7.16×10^{-1}, and 4.54×10^{-1}, respectively. This means that EEDSPSO is not significantly better than HCLPSO, BLPSO, and BFLPSO. But the number of functions in which EEDSPSO outperforms HCLPSO,

Table 5 Wilcoxon sign test in CEC2013 (D=50).

Comparison	p-value	R^+	R^-	$n/w/t/l$
EEDSPSO vs. HCLPSO	5.40×10^{-1}	214	163	28/15/1/12
EEDSPSO vs. GLPSO	2.42×10^{-2}	302	104	28/21/0/7
EEDSPSO vs. BLPSO	7.16×10^{-1}	319	187	28/24/0/4
EEDSPSO vs. MPSO	2.69×10^{-4}	363	43	28/21/0/7
EEDSPSO vs. DSPSO	1.98×10^{-2}	286	92	28/17/0/11
EEDSPSO vs. HCLDMSPSO	1.33×10^{-2}	292	86	28/18/0/10
EEDSPSO vs. BFLPSO	4.54×10^{-1}	205	146	28/19/1/8

Table 6 Wilcoxon sign test in CEC2017 (D=50).

Comparison	p-value	R^+	R^-	$n/w/t/l$
EEDSPSO vs. HCLPSO	3.89×10^{-2}	313	122	29/24/0/5
EEDSPSO vs. GLPSO	5.57×10^{-2}	306	129	29/24/0/5
EEDSPSO vs. BLPSO	3.65×10^{-3}	310	68	29/21/2/6
EEDSPSO vs. MPSO	3.89×10^{-3}	351	84	29/23/0/6
EEDSPSO vs. DSPSO	1.69×10^{-2}	328	107	29/22/0/7
EEDSPSO vs. HCLDMSPSO	1.23×10^{-3}	367	68	29/26/0/3
EEDSPSO vs. BFLPSO	6.34×10^{-4}	375	59	29/24/0/5

Table 7 Wilcoxon sign test in CEC2017 (D=100).

Comparison	p-value	R^+	R^-	$n/w/t/l$
EEDSPSO vs. HCLPSO	5.67×10^{-1}	244	191	29/18/0/11
EEDSPSO vs. GLPSO	2.47×10^{-1}	271	164	29/21/0/8
EEDSPSO vs. BLPSO	2.98×10^{-2}	318	117	29/21/0/8
EEDSPSO vs. MPSO	6.45×10^{-2}	303	132	29/23/0/6
EEDSPSO vs. DSPSO	4.96×10^{-1}	249	186	29/18/0/11
EEDSPSO vs. HCLDMSPSO	6.85×10^{-2}	283	123	29/20/1/8
EEDSPSO vs. BFLPSO	1.44×10^{-1}	285	150	29/20/0/9

BLPSO, and BFLPSO is 15, 24, and 19, respectively. This also indirectly indicates that EEDSPSO outperforms HCLPSO, BLPSO, and BFLPSO.

From Table 6, the significant differences between EEDSPSO and HCLPSO, BLPSO, MPSO, DSPSO, HCLDMSPSO, and BFLPSO are 3.89×10^{-2}, 3.65×10^{-3}, 3.89×10^{-3}, 1.69×10^{-2}, 1.23×10^{-3}, and 6.34×10^{-4}, respectively, which are less than 0.05. This demonstrates that EEDSPSO is significantly better than HCLPSO, BLPSO, MPSO, DSPSO, HCLDMSPSO, and BFLPSO. The number of functions for EEDSPSO superior to GLPSO is 24. This indicates that EEDSPSO is far better than GLPSO.

From Table 7, only the significance level between EEDSPSO and BLPSO is less than 0.05. The positive ranks between EEDSPSO and HCLPSO, GLPSO, MPSO, DSPSO, HCLDMSPSO, and BFLPSO are 244, 271, 303, 249, 283, and 285, respectively. The numbers of functions for EEDSPSO better than HCLPSO, GLPSO, MPSO, DSPSO, HCLDMSPSO, and BFLPSO are 18, 21, 23, 18, 20, and 20, respectively. This clearly indicates the excellent performance of the EEDSPSO algorithm.

In summary, EEDSPSO uses an adaptive framework based on evolutionary experience and a Gaussian crisscross learning strategy, possessing more significant performance relative to other algorithms.

5 Conclusion

To balance the diversity and convergence of the algorithm during the iterative process, this paper proposes an evolutionary experience based multi-strategy particle swarm optimization algorithm. Firstly, the neighborhood topology and the elite archival population are improved using an evolutionary experience based framework. The improved neighborhood topology maintains the population's diversity well. Furthermore, the elite archive is

beneficial to the population convergence performance. And their weight is dynamically changed using adaptive parameters. In addition, the introduction of a Gaussian crisscross learning strategy improves the algorithm's performance on complex problems. Finally, experimental results in CEC2013 (D=50) and CEC2017 (D=50 and 100) show that EEDSPSO outperforms other popular PSO variants. In more complex functions such as hybrid and composition functions, EEDSPSO outperforms other algorithms by a wide margin. This indicates that EEDSPSO has more advantages when facing complex functions. And the average ranking of EEDSPSO in higher dimension 100D is 2.39. The result is further improved relative to that in 50D, which shows that EEDSPSO is expected to try to solve optimization problems in higher dimensions.

In future studies, we will further enhance the diversity of EEDSPSO to enable the entire population to perform effective search behavior. More efficient and novel learning frameworks will be further investigated. Due to the simple implementation and efficiency in exploring global solutions of the proposed algorithm, the solution of many complex problems is always a hot issue. The study of solving complex problems will be beneficial to the optimization of practical applications.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 62066019), Jiangxi Provincial Education Department Project (No. GJJ200819), and Doctoral Startup Foundation of Jiangxi University of Science and Technology (No. 205200100022).

References

[1] J. Kennedy, Swarm intelligence, in *Handbook of Nature-Inspired and Innovative Computing*, A. Y. Zomaya, ed. New York, NY, USA: Springer, 2006, pp. 187–219.

[2] R. Eberhart and J. Kennedy, A new optimizer using particle swarm theory, in *Proc. Sixth Int. Symp. Micro Machine and Human Science*, Nagoya, Japan, 1995, pp. 39–43.

[3] W. Li, Y. Chen, Q. Cai, C. Wang, Y. Huang, and S. Mahmoodi, Dual-stage hybrid learning particle swarm optimization algorithm for global optimization problems, *Complex System Modeling and Simulation*, vol. 2, no. 4, pp. 288–306, 2022.

[4] D. Karaboga and B. Basturk, A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm, *J. Glob. Optim.*, vol. 39, no. 3, pp. 459–471, 2007.

[5] S. Xiao, H. Wang, W. Wang, Z. Huang, X. Zhou, and M. Xu, Artificial bee colony algorithm based on adaptive neighborhood search and Gaussian perturbation, *Appl. Soft Comput.*, vol. 100, p. 106955, 2021.

[6] R. Storn and K. Price, Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[7] W. Li, X. Ye, Y. Huang, and S. Mahmoodi, Adaptive dimensional learning with a tolerance framework for the differential evolution algorithm, *Complex System Modeling and Simulation*, vol. 2, no. 1, pp. 59–77, 2022.

[8] A. Slowik and H. Kwasnicka, Nature inspired methods and their industry applications—Swarm intelligence algorithms, *IEEE Trans. Ind. Inform.*, vol. 14, no. 3, pp. 1004–1015, 2018.

[9] T. Kerdphol, K. Fuji, Y. Mitani, M. Watanabe, and Y. Qudaih, Optimization of a battery energy storage system using particle swarm optimization for stand-alone microgrids, *Int. J. Electr. Power Energy Syst.*, vol. 81, pp. 32–39, 2016.

[10] K. Mahadevan and P. S. Kannan, Comprehensive learning particle swarm optimization for reactive power dispatch, *Appl. Soft Comput.*, vol. 10, no. 2, pp. 641–652, 2010.

[11] A. N. Hussain, A. A. Abdullah, and O. M. Neda, Modified particle swarm optimization for solution of reactive power dispatch, *Res. J. Appl. Sci. Eng. Technol.*, vol. 15, no. 8, pp. 316–327, 2018.

[12] X. -B. Wang, Z. -X. Yang, and X. -A. Yan, Novel particle swarm optimization-based variational mode decomposition method for the fault diagnosis of complex rotating machinery, *IEEE/ASME Trans. Mechatron*., vol. 23, no. 1, pp. 68–79, 2018.

[13] H. Chen, D. L. Fan, L. Fang, W. Huang, J. Huang, C. Cao, L. Yang, Y. He, and L. Zeng, Particle swarm optimization algorithm with mutation operator for particle filter noise reduction in mechanical fault diagnosis, *Int. J. Patt. Recogn. Artif. Intell.*, vol. 34, no. 10, p. 2058012, 2020.

[14] Y. -L. Gao, X. -H. An, and J. -M. Liu, A particle swarm optimization algorithm with logarithm decreasing inertia weight and chaos mutation, in *Proc. 2008 Int. Conf. Computational Intelligence and Security*, Suzhou, China, 2008, pp. 61–65.

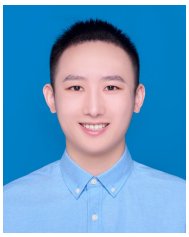
[15] Y. Shi and R. C. Eberhart, Parameter selection in particle swarm optimization, in *Proc. 7th Int. Conf. Evolutionary Programming*, San Diego, CA, USA, 1998, pp. 591–600.

[16] S. -F. Li and C. -Y. Cheng, Particle swarm optimization with fitness adjustment parameters, *Comput. Ind. Eng*., vol. 113, pp. 831–841, 2017.

[17] M. Karimi-Nasab, M. Modarres, and S. M. Seyedhoseini, A self-adaptive PSO for joint lot sizing and job shop scheduling with compressible process times, *Appl. Soft Comput.*, vol. 27, pp. 137–147, 2015.

[18] J. Kennedy and R. Mendes, Population structure and particle swarm performance, in *Proc. 2002 Cong. Evolutionary Computation CEC'02* (*Cat. No. 02TH8600*), Honolulu, HI, USA, 2002, pp. 1671–1676.

[19] J. Kennedy, Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance, in *Proc. 1999 Cong. Evolutionary Computation-CEC99* (*Cat. No. 99TH8406*), Washington, DC, USA, 1999, pp. 1931–1938.
[20] Y. Shi, H. Liu, L. Gao, and G. Zhang, Cellular particle swarm optimization, *Information Sciences*, vol. 181, no. 20, pp. 4460–4493, 2011.
[21] X. H. Shi, Y. H. Lu, C. G. Zhou, H. P. Lee, W. Z. Lin, and Y. C. Liang, Hybrid evolutionary algorithms based on PSO and GA, in *Proc. 2003 Cong. Evolutionary Computation*, Canberra, Australia, 2003, pp. 2393–2399.
[22] B. Yang, Y. Chen, and Z. Zhao, A hybrid evolutionary algorithm by combination of PSO and GA for unconstrained and constrained optimization problems, in *Proc. 2007 IEEE Int. Conf. Control and Automation*, Guangzhou, China, 2007, pp. 166–170.
[23] B. Y. Qu, P. N. Suganthan, and S. Das, A distance-based locally informed particle swarm model for multimodal optimization, *IEEE Trans. Evol. Comput.*, vol. 17, no. 3, pp. 387–402, 2013.
[24] L. Wang, B. Yang, and J. Orchard, Particle swarm optimization using dynamic tournament topology, *Appl. Soft Comput.*, vol. 48, pp. 584–596, 2016.
[25] W. He, W. Gong, L. Wang, X. Yan, and C. Hu, Fuzzy neighborhood-based differential evolution with orientation for nonlinear equation systems, *Knowl. Based Syst.*, vol. 182, p. 104796, 2019.
[26] W. Li, B. Sun, Y. Huang, and S. Mahmoodi, Adaptive complex network topology with fitness distance correlation framework for particle swarm optimization, *Int. J. Intell. Syst.*, vol. 37, no. 8, pp. 5217–5247, 2022.
[27] Y. -H. Zhang, Y. -J. Gong, Y. Gao, H. Wang, and J. Zhang, Parameter-free voronoi neighborhood for evolutionary multimodal optimization, *IEEE Trans. Evol. Comput*., vol. 24, no. 2, pp. 335–349, 2019.
[28] W. Li, Y. Sun, Y. Huang, and J. Yi, An adaptive differential evolution algorithm using fitness distance correlation and neighbourhood-based mutation strategy, *Connect. Sci.*, vol. 34, no. 1, pp. 829–856, 2022.
[29] A. Lin, W. Sun, H. Yu, G. Wu, and H. Tang, Adaptive comprehensive learning particle swarm optimization with cooperative archive, *Appl. Soft Comput.*, vol. 77, pp. 533–546, 2019.
[30] X. Xia, L. Gui, F. Yu, H. Wu, B. Wei, Y. -L. Zhang, and Z. -H. Zhan, Triple archives particle swarm optimization, *IEEE Trans. Cybern*., vol. 50, no. 12, pp. 4862–4875, 2019.
[31] B. Wei, X. Xia, F. Yu, Y. Zhang, X. Xu, H. Wu, L. Gui, and G. He, Multiple adaptive strategies based particle swarm optimization algorithm, *Swarm Evol. Comput.*, vol. 57, p. 100731, 2020.
[32] L. Pan, Y. Zhao, and L. Li, Neighborhood-based particle swarm optimization with discrete crossover for nonlinear equation systems, *Swarm Evol. Comput.*, vol. 69, p. 101019, 2022.
[33] X. Tao, X. Li, W. Chen, T. Liang, Y. Li, J. Guo, and L. Qi, Self-adaptive two roles hybrid learning strategies-based particle swarm optimization, *Inf. Sci.*, vol. 578, pp. 457–481, 2021.
[34] X. Chen, H. Tianfield, and W. Du, Bee-foraging learning particle swarm optimization, *Appl. Soft Comput.*, vol. 102, p. 107134, 2021.
[35] S. Wang, G. Liu, M. Gao, S. Cao, A. Guo, and J. Wang, Heterogeneous comprehensive learning and dynamic multi-swarm particle swarm optimizer with two mutation operators, *Inf. Sci.*, vol. 540, pp. 175–201, 2020.
[36] X. Zhang, X. Wang, Q. Kang, and J. Cheng, Differential mutation and novel social learning particle swarm optimization algorithm, *Inf. Sci.*, vol. 480, pp. 109–129, 2019.
[37] D. Tian and Z. Shi, MPSO: Modified particle swarm optimization and its applications, *Swarm Evol. Comput.*, vol. 41, pp. 49–68, 2018.
[38] X. Chen, H. Tianfield, C. Mei, W. Du, and G. Liu, Biogeography-based learning particle swarm optimization, *Soft Comput.*, vol. 21, no. 24, pp. 7519–7541, 2017.
[39] Y. -J. Gong, J. -J. Li, Y. Zhou, Y. Li, H. S. -H. Chung, Y. H. Shi, and J. Zhang, Genetic learning particle swarm optimization, *IEEE Trans. Cybern*., vol. 46, no. 10, pp. 2277–2290, 2015.
[40] N. Lynn and P. N. Suganthan, Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation, *Swarm Evol. Comput.*, vol. 24, pp. 11–24, 2015.
[41] O. Olorunda and A. P. Engelbrecht, Measuring exploration/exploitation in particle swarms using swarm diversity, in *Proc. 2008 IEEE Cong. Evolutionary Computation* (*IEEE World Cong. Computational Intelligence*), Hong Kong, China, 2008, pp. 1128–1134.



Jianghui Jing received the BEng degree from Zhongyuan University of Science and Technology in 2021. He is currently pursuing the master degree at the School of Information Engineering, Jiangxi University of Science and Technology. His current research interests are evolutionary optimization, evolutionary state estimation, and evolutionary strategy design.

Yangtao Chen received the BEng degree from Jiangxi University of Science and Technology in 2020. He is currently pursuing the master degree at the School of Information Engineering, Jiangxi University of Science and Technology. His main research interest is particle swarm optimization algorithm and multi-objective evolutionary algorithm.

Wei Li received the PhD degree from South China Agricultural University in 2018, the MEng degree in computer application technology from Jiangxi University of Science and Technology in 2008, and the BEng degree in computer science and technology from Jiangxi University of Science and Technology in 2003. He is currently an associate professor at the School of Information Engineering, Jiangxi University of Science and Technology. He has over 30 papers published in fully refereed international journals and conferences and has served as the program chair or program committee member in many international conferences. His research focuses mainly on computational intelligence, evolutionary optimization, fitness landscape analysis, and large-scale optimization.

Xunjun Chen received the PhD degree in computer application technology from Hohai University in 2018. He is currently working at the School of Information Engineering, Jiangxi University of Science and Technology. His research interests include computational intelligence, machine learning security, and privacy computing.

Ata Jahangir Moshayedi received the PhD degree in electronic science in the field of mobile olfaction from Savitribai Phule Pune University, Pune, India in 2015. He is an associate professor at Jiangxi University of Science and Technology, China. He is a member of IEEE, ACM, and International Association of Engineers (IAENG). He is a life member of Instrument Society of India and Speed Society of India. He is a member of the editorial team of various conferences and journals like *International Journal of Robotics and Control*, *Bulletin of Electrical Engineering and Informatics*, *International Journal of Physics and Robotics Applied Electronics*, etc. He published various papers in national journals and conferences and three books. He is the owner of 2 patents and nine copyrights. His research interests include robotics and automation, sensor modeling, biology-inspired robots, mobile robot olfaction, column tracking, embedded systems, machine vision-based systems, virtual reality, machine vision, and artificial intelligence.