Hindawi
Computational Intelligence and Neuroscience
Volume 2020, Article ID 2815802, 10 pages
https://doi.org/10.1155/2020/2815802

Research Article

A Species Conservation-Based Particle Swarm Optimization with Local Search for Dynamic Optimization Problems

Dingcai Shen,[1,2] Bei Qian,[3] and Min Wang[1]

[1] *College of Mathematics and Computer Science, Gannan Normal University, Ganzhou 341000, China*
[2] *School of Computer and Information Science, Hubei Engineer University, Xiaogan 432000, China*
[3] *Finance Office, Gannan Normal University, Ganzhou 341000, China*

Correspondence should be addressed to Dingcai Shen; dcshensa@gmail.com

Received 21 October 2019; Revised 5 July 2020; Accepted 13 July 2020; Published 1 August 2020

Academic Editor: Raşit Köker

In the optimization of problems in dynamic environments, algorithms need to not only find the global optimal solutions in a specific environment but also to continuously track the moving optimal solutions over dynamic environments. To address this requirement, a species conservation-based particle swarm optimization (PSO), combined with a spatial neighbourhood best searching technique, is proposed. This algorithm employs a species conservation technique to save the found optima distributed in the search space, and these saved optima either transferred into the new population or replaced by the better individual within a certain distance in the subsequent evolution. The particles in the population are attracted by its history best and the optimal solution nearby based on the Euclidean distance other than the index-based. An experimental study is conducted based on the moving peaks benchmark to verify the performance of the proposed algorithm in comparison with several state-of-the-art algorithms widely used in dynamic optimization problems. The experimental results show the effectiveness and efficiency of the proposed algorithm for tracking the moving optima in dynamic environments.

1. Introduction

As a very challenging optimization tool, evolutionary algorithms (EAs) have been successfully applied to the optimization problems in static environments. Nevertheless, EAs have not been effectively used to solve optimization problems in dynamic environments, which are very common in many real-world applications, for example, the changes of vehicle routing due to the temporary traffic control or sudden changes in weather, the newly added artefacts in production scheduling, and uncertain market factors lead to changes in financial trading models. In these complex real-world problems, its constraints and coefficients or even objectives may vary with time. The problems with these characters can be modelled as dynamic optimization problems (DOPs). In DOPs, the worse candidate solutions in the past can be the optimal solutions in the new environment, and vice versa. The development of solution strategies in the area of EAs that may work in such uncertain environments raises new challenges.

In static environments, the goal is to find a single optimum or multioptima in the search space. However, in dynamic environments, the goal of the algorithms is no longer to locate the optimal solution, but to continuously track the moving optimum as closely as possible. Many researchers have introduced various strategies into canonical EAs to enhance their ability for tracking changing optima. In static optimization, convergence is a positive factor for the algorithms to locate the global optimum; however, it may weaken the ability of the algorithm to find the moving optimum in a later evolving process because of the diversity loss. In order to increase or maintain the population diversity, researchers have developed many schemes to enhance the canonical EAs' ability to locate moving optimum in dynamic environments.

The simplest method of solving DOPs is to regard each change as the new optimization and reinitialize the population. However, in real-world applications, most of the environmental changes may not be too drastic, and the new optimum will be in some sense related to the historical optimal solutions [1]. In that case, saving the old optimum according to certain strategies is beneficial to the optimization in the new environment. Thus, the memory-based scheme is a widely used method adopted in DOPs [2–4]. In [2], the authors combined the multipopulation scheme and an improved memory strategy, which is saving the elite individual retrieved from each subpopulation, to enhance the exploration ability of the algorithm. In [4], a similar technique as [2] is used for the optimization of DOPs.

A similar scheme to memory named species is also a commonly used technique by EAs' community [5]. Species-based EAs are often regarded as a kind of multiswarm algorithm. It preserves the candidate solutions distributed in the search space according to a predefined radius, and after the evolution in each generation, the saved species seeds are either replacing the worse individual, if there has an unprocessed individual within the search radius, or replacing the worst unprocessed individual in the current population, if there is no solution within the search radius [6]. Many experimental results have shown that multiswarm strategy is helpful in locating multiple optima in multimodal optimization problems, and some researchers have extended this strategy to DOPs and achieved a good performance in locating and tracking the moving optimum [7, 8].

In EAs' community, there are two well-known strategies, which are also commonly adopted in the optimization of DOPs to promoting diversity of the population, named hypermutation [9] and random immigrants. Hypermutation increases population diversity by drastically increasing the mutation rate as the environment changes, and random immigrants strategy replaces a fraction of EAs' population at each generation. The selection of individuals to be replaced often has two ways: one is randomly selected to be replaced and this way seems to be losing the good candidates in some cases, and another one is replacing the worst individuals in the current population.

Particle swarm optimization (PSO) [10] is a population-based stochastic optimization algorithm; it can be considered as one of the most popular nature-inspired metaheuristics for continuous optimization, which was first developed by Eberhart and Kennedy in 1995. Due to its simple concept and easy implementation, PSO has been developed rapidly in the last two decades [11]. PSO has been widely used in the optimization in static environments, and many important research studies have been achieved [12–14]. In recent years, with more and more attention being paid to the research area of DOPs, PSO has been widely applied to the optimization of DOPs.

According to the characteristics of DOPs, and the commonly adopted strategies introduced mentioned above, we observed that the population diversity and historical information play an important role in the optimization of DOPs. Motivated by these observations, a species conservation combined with a spatial neighbourhood best searching strategy is integrated into canonical PSO (denoted as sslPSO) during the evolutionary process, to strengthen the exploration and exploitation ability of the PSO. In each iteration, in order to mitigate the loss of population diversity, the best individual in each subswarm is archived, and then, these species seeds are transferred into the next generation. Experiments on a commonly used benchmark of moving peaks benchmark (MPB) are carried out to investigate the effect of our proposed algorithm. The experimental results show that the proposed algorithm has a promising performance in solving the DOPs.

The primary contribution of sslPSO is an enhanced species conservation, which uses species conserving during the environmental change period; at the same time, the spatial neighbourhood searching is introduced in the updating procedure. The comparison results indicate that the introduction of species conservation is beneficial for the tracking of moving optima in dynamic environments. Meanwhile, the spatially local best searching in the updating of position ensures the algorithm's high exploration ability while maintaining high exploitation ability.

The rest of this paper is arranged as follows. In Section 2, the canonical PSO and its application in DOPs and some related works included in this study are briefly reviewed. The proposed sslPSO algorithm is described in detail in Section 3. Section 4 compares the sslPSO with other state-of-the-art algorithms widely used in DOPs which are presented to show the effectiveness and efficiency of the proposed algorithm. Finally, Section 5 concludes the study and outlines future work.

2. Background and Related Works

In this section, we first give a brief description of the definition of DOP. After that, the PSO framework and its application in DOPs and the need of species in dynamic environments are given briefly.

2.1. DOP. Dynamic optimization problems are optimization problems in which one or more of its problem parameters, such as objective function, constraints, or environmental coefficients, may change over time. The dynamic character of the problem makes the optimization procedure much more complicated than the static optimization problem.

A DOP can be defined in general as follows:

$$\begin{cases} \text{opt} & f(\mathbf{x}, t), \\ \text{s.t.} & g_i(\mathbf{x}, t) \leq 0, \quad (i = 1, 2, \ldots, m), \\ & h_j(\mathbf{x}, t) = 0, \quad (j = 1, 2, \ldots, n), \end{cases} \tag{1}$$

where $f(\mathbf{x}, t)$is the objective function of the problem. $\mathbf{x}$ is the decision vector of Ddimensions in the search space, and $g_i(\mathbf{x}, t)$ and $h_j(\mathbf{x}, t)$ denote the ith inequality constraint and jth equality constraint of the problem, respectively. Both of them may vary during the evolutionary process for specific change type in real-world applications. In our study, we only consider the dynamic optimization without any dynamic constraints.

2.2. Particle Swarm Optimization (PSO). In a PSO algorithm, each particle i is a potential solution in the D-dimensional search space. The position of ith particle in a population with N individuals $(\mathbf{X}_1, \mathbf{X}_2, \ldots \mathbf{X}_N)$ can be represented by $\mathbf{X}_i = (X_{i1}, X_{i2}, \ldots, X_{i\quad D})$, and the velocity of this particle is represented by $\mathbf{V}_i = (V_{i1}, V_{i2}, \ldots, V_{iD})$. In a commonly used update model, each particle is updated by two best solutions, $\mathbf{P}_i = (P_{i1}, P_{i2}, \ldots, P_{i\quad D})$, the best position of particle i it has experienced, and $\mathbf{P}_g = (P_{g1}, P_{g2}, \ldots, P_{gD})$, the best position discovered so far by all particles. At the beginning, N particles are randomly generated over the search space. Then, all particles search for the optimum by fly over the search space, until the global optimal position is found. At each iteration, the new velocity and position of each particle can be updated according to its current velocity and position as follows [10]:

$$\begin{aligned}\mathbf{V}_i(t+1) = \omega\mathbf{V}_i(t) + \varphi_1\left(\mathbf{P}_i(t) - \mathbf{X}_i(t)\right) \\ + \varphi_2\left(\mathbf{V}_g(t) - \mathbf{X}_i(t)\right),\end{aligned} \tag{2}$$

$$\mathbf{X}_i(t+1) = \mathbf{X}_i(t) + \mathbf{V}_i(t+1), \tag{3}$$

where

$$\begin{aligned}\varphi_1 &= c_1\mathbf{R}_1, \\ \varphi_2 &= c_2\mathbf{R}_2,\end{aligned} \tag{4}$$

ω is a parameter used to control the influence degree of the previous velocity to the current velocity, which is named inertia weight. c_1 and c_2 are two positive constants balancing the contribution of the particle's own previous best position and the best position the whole swarm had attained. $\mathbf{R}_1$ and $\mathbf{R}_2$ are two D-dimensional random vectors uniformly distributed in the interval $U(0.0, 1.0)$.

In the velocity updating procedure using equation (2), all particles are attracted to the best solution $(P_g(t))$ found by all members of the population. This updating model is typically called global best (denoted as gbest). In the gbest model, every individual learns from its own experience and imitates the very best member found in the population [15]. Another commonly used model is called local best (denoted as lbest). In the lbest network, each individual's movement is attracted by the best performance of its neighbours in a certain range. The determination of neighbours usually has two ways: one is determined according to the index which is randomly assigned at the initialization step and keeps constant in the whole evolution process; the other is determined on the basis of spatial distance between particles (the difference between these two structures will be described in detail in Section 3). The gbest model is generally considered to have a faster convergence rate; in the meantime, it would be more likely to get stuck in local optimal solution. The lbest model, however, is converged slower with less likely to be trapped into the local optimum than that of gbest model. The PSO algorithm repeatedly applies the update procedure until the maximum evaluations are reached. The pseudocode of the canonical PSO algorithm is described in Algorithm 1.

2.3. PSO in Dynamic Optimization Problems. The application of PSO to dynamic optimization problems has been widely studied in recent years [5, 16–20]. Similar to other EAs, there are two key facts that must be faced in the application of PSO to dynamic environment: one is the outdated memory, and the other is the loss of diversity [18]. Most of the existing research studies introduce various strategies into canonical PSO to overcome these two problems. When a new environment comes, the previous best location visited by the particle becomes outdated. In this case, a simple and effective response is to recalculating the fitness value of the objective function of each particle. However, in real-world applications, most of the environmental changes are usually not dramatic. When the dynamic problem changes periodically or recurrently, it might be helpful to reuse previously found solutions to reduce the computation times [21]. Many researchers adopt various strategies to save redundant information to the memory particles for later use in case of environmental changes. In [22], a triggered memory scheme is proposed, in which the best individuals found by the "explore" population are stored in the memory, and two memory retrieval schemes, named memory-based resetting and memory-based immigrants, are adopted to retrieve memory when environmental change is detected. Zhu et al. [4] propose a new memory scheme with a large memory capacity to improve the performance of multipopulation algorithms for DOPs.

In dynamic optimization, convergence is unfavourable to the performance of the algorithms. Therefore, many researchers have introduced various schemes into canonical PSO to maintain or increase the diversity of the population during evolution. Inspired by the movement of atom, Blackwell et al. [23, 24] proposed charged PSOs to maintaining diversity in the solving of DOPs. Multiple swarm/population methods are often considered as another effective strategy for increasing population diversity in the optimization of DOPs. In [25], the authors proposed two multiswarm paradigms of particle swarms and verified the effectiveness on a widely used moving peaks benchmark. Li et al. [8] proposed an adaptive multiswarm optimizer to enable adaptively to change the number of populations in dynamic environments.

2.4. The Need of Species Conservation. In biology, species are a group of creatures with similar characteristics; similarly, in EAs' community, a species represents a collection of individuals with common characteristics. In all similarity measurements, the Euclidean distance is the most commonly used similarity measurement. The smaller the distance between two individuals, the more similar they are.

The distance between two individuals $\mathbf{X}_i = (X_{i1}, X_{i2}, \ldots, X_{iD})$ and $\mathbf{X}_j = (X_{j1}, X_{j2}, \ldots, X_{jD})$ is defined as follows:

$$d\left(X_i, X_j\right) = \sqrt{\sum_{k=1}^{D}\left(X_{i,k} - X_{j,k}\right)^2}. \tag{5}$$

```
Randomly initialize $N$ particles of population $P$within the problem's search space;
Evaluate each individual in $P$;
**for each** $P_i \in P$**do**
    $P_i^{\text{best}} = P_i$;
**end for**
Find the global best particle $P_g^{\text{best}}$;
**while** termination condition is not reached **do**
    Apply equation (2) to update each particle's velocity;
    Apply equation (3) to update each particle's location;
    Calculate the fitness value of $P_i$;
    **if** $f(P_i)$ is better than $f(P_i^{\text{best}})$
        $P_i^{\text{best}} = P_i$;
    **end if**
    **if** $f(P_i)$ is better than $f(P_g^{\text{best}})$
        $P_g^{\text{best}} = P_i$;
    **end if**
**end while**
```

Algorithm 1: Pseudocode of the canonical PSO algorithm.

In this paper, the above definition of distance is used to measure the similarity between two individuals.

Figure 1 illustrates the need for species conservation during evolution. As Figure 1 shows, after some iterations of the EA, most of the individuals have converged on peak S_1, and at the same time, an individual X_3 with low fitness has emerged. Because of its low fitness, the individual X_3 will have a high probability of being eliminated in the next generation. However, this individual is very important in the multimodal optimization problems that need the algorithms to find all the optima, or in dynamic optimization problems that need the algorithms to track the moving optima. In order for the individual to survive in the next generation, this individual must be preserved. The species conservation procedure is described in Algorithm 2.

3. The Proposed Algorithm

Our proposed algorithm, denoted by sslPSO, takes advantage of the memory strategy to enhance the tracking ability of the PSO in dynamic environments. After initialization, the proposed enhanced species conservation-based PSO algorithm conducts the main loop with four main stages: species determination, particle update, seeds conservation, and particle attractor and swarm attractor update. These stages are briefly described as follows.

The species-based scheme is introduced into the multimodal optimization or dynamic optimization [17], in which the whole population is divided into several species according to their similarity. In the later of the evolution process, most individuals are concentrated in a few optima, resulting in too large number of individuals in the corresponding species and a rapid decline in population diversity. Thus, a mechanism for preventing too many individuals following a single peak is needed for the algorithm to track the moving optima. As in [17], a parameter $PS_{\max}$ is also introduced in our study. If the number of particles in a species exceeds $PS_{\max}$, only the pre-$PS_{\max}$ fittest particles are retained in the species, and the remainder will be randomly reinitialized, thus to maintain the population diversity during the evolution.

Following the species determination, the update procedure is applied. In this study, an improved lbest-based update strategy is introduced. In the usual lbest topology, each particle is affected by the best performance of its r immediate neighbours. However, the vector indices are assigned as their generating order at the initialization step and are kept constant throughout the evolution. The indices-based neighbours may belong to a different group far apart spatially in the search space, resulting in a degradation of the exploitation ability of the algorithm.

As Figure 2(a) shows, the individuals $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3$, and $\mathbf{X}_4$ are neighbours to each other, but it can be seen from the figure that some of them are spatially far apart. Take $\mathbf{X}_1$ as an example, if an index-based neighbour is used, then $\mathbf{X}_2$ or $\mathbf{X}_3$ will be selected as the local best particle, and $\mathbf{X}_1$ searches around $\mathbf{X}_2$ or $\mathbf{X}_3$, resulting in severe decrease in the exploitation ability of the algorithm. Different from the indices-based neighbourhood, we use spatial-based neighbourhood in the update procedure, and the parameter r is renamed as *neighbourhood radius* and denoted by n_r, which is used to search the fittest individual in what range. The determination of local best in n_r neighbours is as shown in Algorithm 3. From Figure 2(b), we can see that the update of $\mathbf{X}_1$ is only affected by the individuals around it in a certain range. If the radius is small, i.e., n_{r1}, then the update of $\mathbf{X}_1$ is affected by 4 individuals (denoted by the solid blue circle); if the radius is further increased, i.e., $(n_{r1} < n_{r2})$, then the update of the $\mathbf{X}_1$ will be affected by 7 individuals around it. The difference in neighbourhood radius will affect the individual's exploitation ability and exploration ability. If the neighbourhood radius is small, the algorithm's exploitation ability will be higher; otherwise, the algorithm's exploration ability will increase. In dynamic environments, how to achieve a balance between exploitation ability and exploration ability is an important fact for the algorithm to

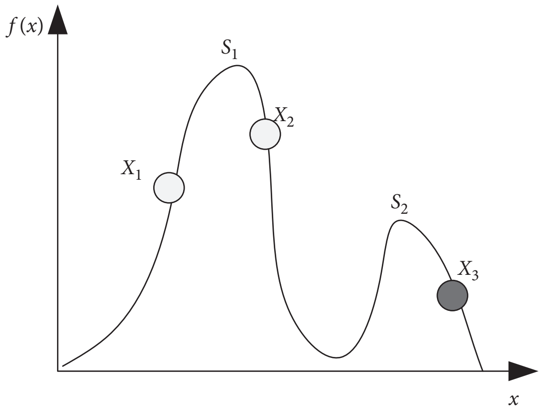
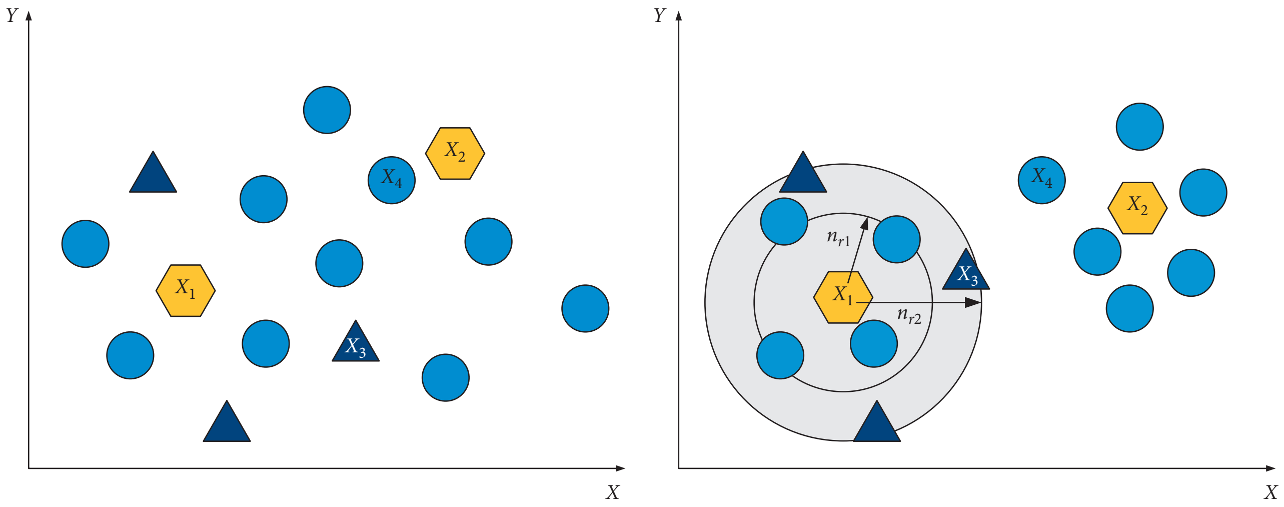


Figure 1: An illustration of the need for species conservation.

```
Input: species seed set $P_{\text{seeds}}$ saved at the species seed determination procedure and the newly generated population $P^{t+1}$, species radius $\delta_s$.
Output: the population after the species conservation procedure.
Mark all individuals in population $P^{t+1}$ as unprocessed;
for all $x \in P_{\text{seeds}}$ do
    Select the worst unprocessed individual $p_x \in P^{t+1}$ which satisfy $\mathrm{d}(p_x, x) \leq \delta_s$;
    if $p_x$ exists then
        if $f(x)$ is fitter than $f(p_x)$ then
            replace $p_x$ with $x$;
        end if
    else
        replace the worst unprocessed $p_x$ in $P^{t+1}$ with $x$;
    end if
    Mark $p_x$ as processed;
end for
```

Algorithm 2: The procedure of species conservation.



(a)

(b)

Figure 2: The illustration of different neighbourhood structures: (a) indices-based neighbourhood and (b) spatial-based neighbourhood.

```
Input: particle $p_i$ of population $P$ with size $N$
Output: particle $p_i^{l\text{best}}$ in the $n_r$ neighbourhoods of $p_i$
for $j \longleftarrow 1$ to $N$ do
    Calculate Euclidean distance $\mathrm{d}(p_i, p_j), i \neq j$;
end for
Sort $p_i$'s neighbours set $P_{rs}^i$ in ascending order according to the $d(p_i, p_j)$;
$P_{rs}^i = \{p_1^{rs}, p_2^{rs}, \ldots, p_{N-1}^{rs}\}$;
Select the formerly ranked $n_r$ particles in $P_{rs}^i$;
$P_r = \left\{p_1^{rs}, p_2^{rs}, \ldots, p_{n_r}^{rs}\right\}$;
Search the local best $p_i^{l\text{best}}$ in $P_r$.
```

Algorithm 3: The algorithm of spatially neighbourhoods best searching.

improve its performance. The performance of the algorithm under different neighbourhood radius will be analysed later in this paper.

In order to accelerate the convergence process over an environment period, the inertia weight ω in equation (2) is linearly decreasing from $\omega_{\max}$ to $\omega_{\min}$ as follows:

$$\omega = \omega_{\max} - \frac{\left(\omega_{\max} - \omega_{\min}\right) \times E_{\text{itr}}}{E_{\text{itr}}}, \tag{6}$$

where $\omega_{\max}$ and $\omega_{\min}$ are the maximum and minimum values of ω, respectively, and E_{itr} is the iteration counter during an environment cycle (in this study, it is assumed that the environmental change cycle is known in advance).

Once all the particles are updated, some species may not survive. Thus, seeds conservation process is conducted immediately after the update procedure. The species seeds are either copied to the new population or substituted by better samples of the same species.

Algorithm 4 describes the framework of the sslPSO and Figure 3 presents the flowchart of the algorithm.

4. Experimental Evaluation

In this section, a series experiments are carried out based on the moving peaks benchmark problem to measure the effectiveness of the proposed algorithm (run on an Intel Core i5 4590@ 3.30 GHz processor with 8 Gb RAM on Windows 10 Home Premium 64-bit operation system), and then, the performance of sslPSO is compared with a set of EAs taken from the literature for DOPs. The involved algorithms include CPSO [19], mCPSO [24], mQSO [24], SPSO [26], and rSPSO [27] and canonical PSO. All the results of the peer algorithms presented in this paper are taken from the paper where they were proposed. In the following section, we will introduce the benchmark function adopted in our experiments, as well as the performance measure and parameter settings. Finally, we present the experimental results and analysis.

4.1. Benchmark Problem. The moving peaks benchmark (MPB) was proposed by Branke [28] and has been widely used to test the performance of the dynamic optimization algorithms. Within an MPB problem with N_p peaks in a D-dimensional search space, the location, height, and the width of the peaks can be varied at a certain frequency. The form of the canonical MPB is formulated as follows:

$$F(\mathbf{X}, t) = \max_{i=1,\ldots,N_p} \frac{H_i(t)}{1 + W_i(t) \sum_{d=1}^{D} \left(X_d(t) - X_{id}(t)\right)^2}, \tag{7}$$

where $W_i(t)$ and $H_i(t)$ denote the width and height of peak i at time t, respectively, and $X_{id}(t)$ denotes the dth element of peak i at time t, and $X_d(t)$ is the dth element of particle $\mathbf{X}$ at time t.

The location of each peak is shifted by a vector v of a fixed distance s in a random direction. The parameter s is named as the shift length, which is used to define the change severity of the dynamic problems. The move of a single peak can be defined as follows:

$$\mathbf{v}_i(t) = \frac{s}{\left|\mathbf{r} + \mathbf{v}_i(t-1)\right|}\left((1-\lambda)\mathbf{r} + \lambda \mathbf{v}_i(t-1)\right), \tag{8}$$

where $\mathbf{r}$ is a random vector and λ is used to determine the move direction correlated to the previous movement, $\lambda = 0$ for a random direction and $\lambda > 0$ for a direction related to the previous direction.

More formally, a move of a single peak can be denoted as follows:

$$H_i(t) = H_i(t-1) + \text{height}_{\text{severity}} \times \sigma, \tag{9}$$

$$W_i(t) = W_i(t-1) + \text{width}_{\text{severity}} \times \sigma, \tag{10}$$

$$X_i(t) = X_i(t-1) + v_i(t), \tag{11}$$

$$\sigma \in N(0, 1). \tag{12}$$

4.2. Performance Measurement. Properly measuring the performance of the algorithms is vital in the optimization of DOPs. There are several most commonly used criteria to evaluate the algorithms in existing studies. Existing performance measures in DOPs can be classified into two main groups: optimality-based and behaviour-based. Interested readers can refer to [21] for a more detailed description. In this study, in order to quantify the performance of the proposed algorithm (sslPSO) within a dynamic environment, and for a fair comparison with the peer algorithms, we

```
Initialize the population $P$ with size $N$;
**While** stop criterion is not satisfied **do**
    Species determination;
    **for all** $i$ such that $0 \leq i \leq N$
        Search the local best $P_i^{l\text{best}}$ with Algorithm 3;
        Apply equation (2) to update $V_i$(replace $P_g(t)$ with $P_i^{l\text{best}}$);
        Apply equation (3) to update $X_i$;
    **end for**
    Seeds conservation;
    Update each particle's attractor and the population's attractor;
**end while**
```

Algorithm 4: The framework of sslPSO.

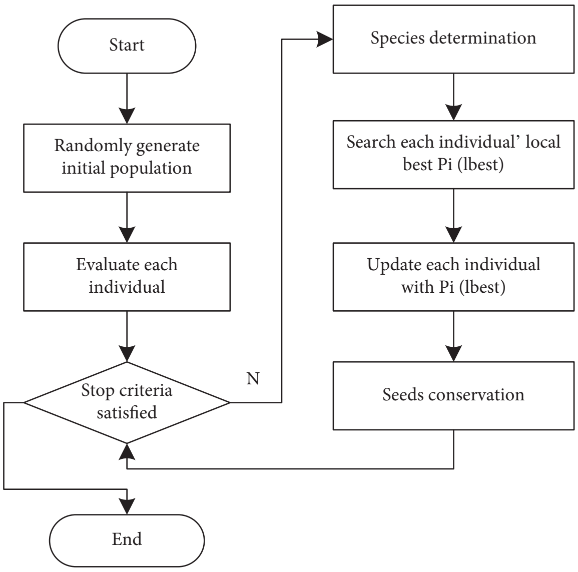

Start
Randomly generate initial population
Evaluate each individual
Stop criteria satisfied
N
Y
End
Species determination
Search each individual' local best Pi (lbest)
Update each individual with Pi (lbest)
Seeds conservation

Figure 3: Flowchart of sslPSO.

used the MPB as the test suite, and the offline error measurement is adopted, which is defined as follows:

$$e_{\text{off}} = \frac{1}{N_e} \sum_{k=1}^{N_e} \left(f_k^{\text{opt}} - f_k \right), \tag{13}$$

where f_k is the best evaluation found by an algorithm right before the kth environmental change, f_k^{opt} is the theoretical optimum value of the kth environment, e_{off} is the average of all differences between f_k^{opt} and f_k over the environmental changes, and N_e is the total number of environmental changes in a run.

4.3. Parameter Settings. The default settings of the MPB used in the experiments in this paper are given in Table 1, which are the same as in all the peer algorithms (for SPSO in [26], the authors in [27] tuned the algorithms to optimising this benchmark, and the results are taken from [27]). In Table 1, the term "change of frequency, U" means that for every U fitness evaluation, an environment change will occur. Initially, P peaks are randomly generated with the given boundaries of position, height, and width as shown in Table 1. The height and width are shifted randomly with the shift severity s in the range $H = [30, 70]$ and $W = [1, 12]$, respectively.

In our sslPSO algorithm, the population size is set to 100, and the learning factors c_1 and c_2 are both set to 1.7. The inertia weight ω is initially set to $\omega_{\max} = 0.9$ and then decreases linearly to $\omega_{\min} = 0.3$ over the entire change cycle. The other parameters are set as follows: the species size is confined to $\text{PS}_{\max} = 10$, the species distance σ_s is set to 30, the *neighbourhood radius* is set to $n_r = 3.0$, and the performance with different sizes is also investigated in our study. For each test case, there were $N_e = 100$ environmental changes, which result in $N_e \times U = 100 \times 5{,}000$ fitness evaluations in each run. The result of each experiment is the average of 30 independent runs with different random seeds.

4.4. Experimental Results and Analysis. In this section, the performance of sslPSO is investigated in several aspects, including the effect of neighbourhood radius, the effect of

Table 1: Default settings for the MPB problem.

Parameter	Value
Number of peaks, P	10
Change of frequency, U	5000
Height severity	7.0
Width severity	1.0
Peak shape	Cone
Shift length, s	1.0
Number of dimensions, D	5
Search space range	[0, 100]
Peak height, H	[30, 70]
Peak width, W	[1, 12]
Correlation coefficient, λ	0

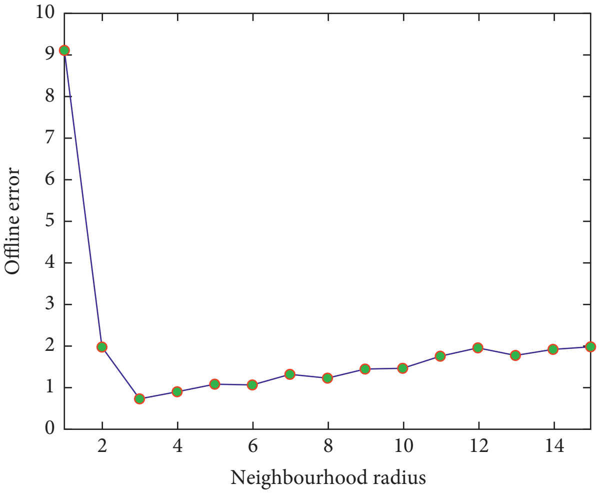


Figure 4: Offline error of sslPSO on the MPB problem with different neighbourhood radius.

varying the shift severity, and the ability of locating and tracking the moving optima, respectively.

4.4.1. Sensitivity Analysis of Parameter n_r. Figure 4 presents the offline error achieved by sslPSO on the MPB problem with different neighbourhood radius. From Figure 4, we can see that when the particles learn from only its nearest neighbour (i.e., the neighbourhood radius is set to 1.0), the sslPSO achieved the largest offline error. With the increase of neighbourhood radius, the offline error begins to decrease. When the neighbourhood radius reaches 3.0, the optimal solution obtained is 0.75 and then increases slowly with the increase of the neighbourhood radius. When the neighbourhood radius is equal to the popsize, the lbest model is equivalent to gbest model, and the offline error obtained by the algorithm is 6.56 (not plotted in Figure 4).

As can be seen from Figure 4, when the neighbourhood radius is small, the algorithm has a good balance of exploration ability and exploitation ability. As the neighbourhood radius increases, the individual moves closer to the global optima at a faster rate, which weakens the exploration ability of the algorithm. And this is the key issue that needs to be overcome in the optimization problem in the dynamic environments.

4.4.2. Comparison of sslPSO with Peer Algorithms. In this section, a set of experiments is conducted to compare the performance of sslPSO with peer algorithms on the MPB problems with different settings of the shift severity parameter*s*. Table 2 presents the experimental results regarding the offline error and standard deviation. The experimental results of the peer algorithms are taken from the corresponding research studies, and the parameters are set to the optimal values which enable them to achieve their best performance (e.g., using the optimal configuration of C(70, 3) for CPSO and using the optimal $n_r = 3.0$ for sslPSO).

From Table 2, it can be seen that the results achieved by sslPSO are much better than the results of the other five algorithms on the MPB problems with different shift severities and are better than the results of the CPSO in most cases, or at least as good as CPSO. As we have speculated, the performance obtained by the canonical PSO is the worst among all algorithms, that is, the canonical PSO almost loses the ability to track moving optima in a dynamic

Table 2: Offline error of algorithms on the MPB problems with different shift severities.

s	sslPSO	CPSO	mCPSO	mQSO	rSPSO	SPSO	PSO
0.0	0.65	0.80	1.18	1.18	0.74	0.95	15.47
	± 0.22	± 0.21	± 0.08	± 0.08	± 0.08	± 0.09	± 2.29
1.0	0.75	1.06	2.05	1.75	1.50	2.51	16.75
	± 0.25	± 0.24	± 0.07	± 0.06	± 0.08	± 0.09	± 2.89
2.0	1.18	1.17	2.80	2.40	1.87	3.78	14.91
	± 0.28	± 0.22	± 0.07	± 0.06	± 0.05	± 0.09	± 3.24
3.0	1.32	1.36	3.57	3.00	2.40	4.96	15.45
	± 0.24	± 0.28	± 0.08	± 0.06	± 0.08	± 0.12	± 3.32
4.0	1.59	1.38	4.18	3.59	2.90	2.56	15.42
	± 0.22	± 0.29	± 0.09	± 0.10	± 0.08	± 0.13	± 3.42
5.0	1.59	1.58	4.89	4.24	3.25	6.76	16.36
	± 0.24	± 0.32	± 0.11	± 0.10	± 0.09	± 0.15	± 4.23
6.0	1.92	1.53	5.53	4.79	3.86	7.68	15.35
	± 0.22	± 0.29	± 0.13	± 0.10	± 0.11	± 0.16	± 4.52

environment. As we know, with the increasing of the shift severity, the ability of the algorithm to track the moving peaks decreases simultaneously. From the results of Table 2, we can see that the performance of all the algorithms degrades when the shift severity increases (there is an exception that canonical PSO achieves the similar worst performance under different shift severities). The offline error of SPSO increases fastest in comparison with the other five algorithms, while the sslPSO and CPSO are two algorithms which are slightly affected by the increase of the shift severity. When the shift severity is set to 0.0, i.e., there is no movement of the peaks, the offline error of the sslPSO achieved the best with 0.65. When the shift severity is set to 1.0, a value setting as most researches adopted, the sslPSO achieved a value of 0.75, which is much better than the other peer algorithms. When the shift severity reaches 6.0, a value which is usually hard for an algorithm to track the moving optima, the offline error of sslPSO is 1.92, which is only slightly higher than that of CPSO, and is much better than the results of the other five algorithms. The results show that sslPSO is very robust to track the moving optima in severely change environments.

5. Conclusions

Particle swarm optimization algorithms have been widely used in the optimization in static environments, and some promising results have been achieved in recent years when it was applied to address DOPs. For DOPs, in order to effectively track the moving optima in dynamic environments, it is usually important to introduce additional strategies to increase or maintain the population diversity or to effectively use the history optima information in the following evolution.

In this work, a species conservation-based PSO combined with a spatial neighbourhood best searching is proposed for DOPs. In order to effectively track the moving optima in dynamic environments, the previously found optima distributed in the population are reserved according to their dissimilarity based on Euclidean distance and are either preserved or replaced by the better individuals within a predefined range in the following evolution. Experimental results on a commonly used benchmark function for DOPs show that the proposed algorithm can greatly improve the performance of PSO in terms of tracking the moving optima in a dynamic fitness landscape with multiple changing peaks. The performance of sslPSO has good expansibility regarding the change severity in the peaks movement in comparison with other peer algorithms. sslPSO performs much better than mCPSO, mQSO, rSPSO, SPSO, and PSO in tracking the moving optima in dynamic environments with different change severities and is better than CPSO or as good as it in each circumstance. When the change severity is small, sslPSO outperforms all the other peer algorithms. In future work, we will consider using fewer memory optima to reduce the computational complexity caused by the participation of the previous optima in the subsequent evolution; it would also be very interesting to investigate the performance of the proposed technique under different change periods and change peaks, and applying to the real-world application is also a promising direction.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Educational Commission of Jiangxi Province of China (Grant no. GJJ160932) and the Educational Commission of Hubei Province of China (Grant no. D20142073) and was supported in part by the National Nature Science Foundation of China (no. 61601128).

References

[1] J. Branke, "Evolutionary approaches to dynamic optimization problems-updated survey," in *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*,

J. Branke and T. B¨ack, Eds., pp. 2–6, San Francisco, California, USA, 2001.

[2] W. Wu, D. Xie, and L. Liu, "Heterogeneous differential evolution with memory enhanced brownian and quantum individuals for dynamic optimization problems," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 32, no. 02, Article ID 1859003, 2018.

[3] A. Sim̃oes and E. Costa, "The influence of population and memory sizes on the evolutionary algorithm's performance for dynamic environments," *Applications of Evolutionary Computing*, vol. 5484, pp. 705–714, 2009.

[4] T. Zhu, W. Luo, and L. Yue, "Combining multipopulation evolutionary algorithms with memory for dynamic optimization problems," in *Proceedings of the 2014 IEEE Congress on Evolutionary Computation*, pp. 2047–2054, IEEE CEC, Beijing, China, July 2014.

[5] W. Luo, J. Sun, C. Bu, and H. Liang, "Species-based Particle Swarm Optimizer enhanced by memory for dynamic optimization," *Applied Soft Computing*, vol. 47, pp. 130–140, 2016.

[6] J.-P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson, "A species conserving genetic algorithm for multimodal function optimization," *Evolutionary Computation*, vol. 10, no. 3, pp. 207–234, 2002.

[7] W. Ye, W. Feng, and S. Fan, "A novel multi-swarm particle swarm optimization with dynamic learning strategy," *Applied Soft Computing*, vol. 61, pp. 832–843, dec 2017.

[8] C. Li, S. Yang, and M. Yang, "An adaptive multi-swarm optimizer for dynamic optimization problems," *Evolutionary Computation*, vol. 22, no. 4, pp. 559–594, dec 2014.

[9] H. G. Cobb, "An investigation into the use of hypermutation as an adaptative operator in genetic algorithms having continuous," Tech. Rep. AIC-90-001, Naval Research Laboratory, Washington, USA, 1990.

[10] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43, IEEE, Nagoya, Japan, October 1995.

[11] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," *Soft Computing*, vol. 22, no. 2, pp. 387–408, 2018.

[12] B. Jana, S. Mitra, and S. Acharyya, "Repository and mutation based particle swarm optimization (rmpso): a new PSO variant applied to reconstruction of gene regulatory network," *Applied Soft Computing*, vol. 74, pp. 330–355, 2019.

[13] N. A. Al-Thanoon, O. S. Qasim, and Z. Y. Algamal, "A new hybrid firefly algorithm and particle swarm optimization for tuning parameter estimation in penalized support vector machine with application in chemometrics," *Chemometrics and Intelligent Laboratory Systems*, vol. 184, pp. 142–152, 2018.

[14] K. Chen, F. Zhou, and A. Liu, "Chaotic dynamic weight particle swarm optimization for numerical function optimization," *Knowledge-Based Systems*, vol. 139, pp. 23–40, 2018.

[15] J. Kennedy and R. C. Eberhart, *Swarm Intelligence, Ser. The Morgan Kaufmann Series in Evolutionary Computation*, Morgan Kaufmann Publishers, Burlington, MA, USA, 2001.

[16] G. Pampar`a and A. P. Engelbrecht, "Self-adaptive quantum particle swarm optimization for dynamic environments," *International Series in Operations Research and Management Science*, vol. 272, pp. 163–175, 2018.

[17] D. Parrott and X. Li, "A particle swarm model for tracking multiple peaks in a dynamic environment using speciation," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 98–103, IEEE, Portland, OR, USA, June 2004.

[18] T. Blackwell, "Particle swarm optimization in dynamic environments," Edited by S. Yang, Y.-S. Ong, and Y. Jin, Eds., in *Evolutionary Computation in Dynamic and Uncertain Environments*, vol. 51, pp. 29–49, Springer, Berlin, Heidelberg, Germany, 2007.

[19] S. Yang and C. Li, "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 6, pp. 959–974, dec 2010.

[20] A. Aboud, R. Fdhila, and A. M. Alimi, "Dynamic multi objective particle swarm optimization based on a new environment change detection strategy," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10637, pp. 258–268, 2017.

[21] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: a survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, pp. 1–24, 2012.

[22] H. Wang, D. Wang, and S. Yang, *Triggered Memory-Based Swarm Optimization in Dynamic Environments, in Applications of Evolutinary Computing*, pp. 637–646, Springer, Berlin, Heidelberg, Germany, 2007.

[23] T. M. Blackwell and P. J. Bentley, "Dynamic search with charged swarms," in *Proceedings of the Genetic and Evolutionary*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 19–26, July 2002.

[24] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anticonvergence in dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 459–472, aug 2006.

[25] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in *EvoWorkshops, ser. Lecture Notes in Computer Science*, G. Raidl, S. Cagnoni, J. Branke et al., Eds., pp. 489–500, Springer, Berlin, Heidelberg, Germany, 2004.

[26] D. Parrott and X. Xiaodong Li, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 440–458, 2006.

[27] S. Bird and X. Li, "Using regression to improve local convergence," in *Proceedings of 2007 IEEE Congress on Evolutionary Computation*, pp. 592–599, IEEE, Singapore, Singapore, September 2007.

[28] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 3, pp. 1875–1882, IEEE, Washington, DC, USA, 1999.