Research Article
Multispecies Coevolution Particle Swarm Optimization Based on Previous Search History

Danping Wang,1,2,3 Kunyuan Hu,1 Lianbo Ma,1,2 Maowei He,1,2,4 and Hanning Chen4

1Department of Information Service & Intelligent Control, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China
2University of Chinese Academy of Sciences, Beijing 100039, China
3Shenyang University, Shenyang 110044, China
4School of Computer Science and Software, Tianjin Polytechnic University, Tianjin 300387, China

Correspondence should be addressed to Maowei He; hemaowei@hotmail.com and Hanning Chen; perfect_chn@hotmail.com

Received 4 November 2016; Revised 11 April 2017; Accepted 19 April 2017; Published 27 June 2017

Abstract
A hybrid coevolution particle swarm optimization algorithm with dynamic multispecies strategy based on K-means clustering and nonrevisit strategy based on Binary Space Partitioning fitness tree (called MCPSO-PSH) is proposed. Previous search history memorized into the Binary Space Partitioning fitness tree can effectively restrain the individuals’ revisit phenomenon. The whole population is partitioned into several subspecies and cooperative coevolution is realized by an information communication mechanism between subspecies, which can enhance the global search ability of particles and avoid premature convergence to local optimum. To demonstrate the power of the method, comparisons between the proposed algorithm and state-of-the-art algorithms are grouped into two categories: 10 basic benchmark functions (10-dimensional and 30-dimensional), 10 CEC2005 benchmark functions (30-dimensional), and a real-world problem (multilevel image segmentation problems). Experimental results show that MCPSO-PSH displays a competitive performance compared to the other swarm-based or evolutionary algorithms in terms of solution accuracy and statistical tests.

1. Introduction
Solving numerical optimization problems is a challenging research endeavor in many scientific areas. Many optimization techniques and search algorithms have drawn their motivation from evolution and social behavior. These include ant colony optimization [1], genetic algorithms [2], differential evolution [3], particle swarm optimization [4], and artificial immune systems [5]. In swarm intelligence (SI) systems, there are many simple individuals who can interact locally with one another and with their environments. Although such systems are decentralized, local interactions between individuals lead to the emergence of global behavior and properties. All of them have many variants, which have excellent performance. These variants are based on various improvement strategies.

In the last few years, several heuristics have been developed to improve the performance and set up suitable parameters for the PSO algorithm. van den Bergh [6] analyzed the trajectory of particles under different inertia weights and acceleration coefficients. The original structure of PSO model reflects an intuitive idea that a particle takes any position on which fitness value is better than where it currently is as the reference input. However, many experiments have shown that the basic PSO algorithm easily falls into local optima when solving complex multimodal problems with a huge number of local optima.

To overcome this problem, researchers conducted lots of studies. Inspired by the coevolution phenomenon in nature, in Cooperative Particle Swarm Optimization (CPSO) [7], multiple swarms are partitioned uniformly to optimize different components of the solution vector cooperatively. Inspired by this work, a CCPSO integrating the random grouping and adaptive weighting schemes was developed and demonstrated great promise in scaling up PSO on high-dimensional nonseparable problems [8]. In [9], a competitive and cooperative coevolutionary PSO (CCPSO) has considerable
potential for solving complex optimization problems by explicitly modeling the coevolution of competing and cooperating species. In Multiswarm Self-Adaptive and Cooperative Particle Swarm Optimization (MSCPSO) [10], particles in each subswarm share the only global historical best optimum to enhance the cooperative capability. In [11], Adaptive Cooperative Particle Swarm Optimizer (ACPSO) facilitates cooperation technique through the usage of the Learning Automata (LA) algorithm. Coevolutionary particle swarm optimization (PSO) algorithm associated with the artificial immune principle (ICPSO) [12] adopts an improved immune-clonal-selection operator for optimizing the elite subpopulation and a migration scheme for the information exchange between elite subpopulation and normal subpopulations.

Different from the above works with static-swarms strategy, some cooperative PSO variants introduce dynamic multiswarms strategy to enhance the ability of exploring local optima. With the aim of maintaining multiple swarms on different peaks, a clustering PSO (CPSO) proposed in [13] employs a nearest neighbor learning strategy to train particles and a hierarchical clustering method to locate and track multiple optima. In hierarchical cluster-based multispecies particle swarm optimization (HCMSPSO) algorithm [14], a swarm is clustered into multiple species at an upper hierarchical level, and each species is further clustered into multiple subspecies at a lower hierarchical level. And in the lower layer, subspecies within the same species are formed adaptively in each iteration during the particle update. In [15], a dynamic multiswarm particle swarm optimizer (DMS-PSO) merges the harmony search (HS) algorithm into each subswarm. These subswarms are regrouped frequently and information is exchanged among the particles in the whole swarm. In a novel optimizer using Adaptive Heterogeneous Particle Swarms (AHPS) [16], to best take advantage of the heterogeneous learning strategies, an adaptive competition strategy is designed for dynamically adjusting the population size of an independent swarm with comprehensive learning and another one with subgradient learning based on their group performance. Roles of learning swarm and learnt swarm in particle swarm optimization with interswarm interactive learning strategy (IILPSO) [17] swap to maintain the diversity when interswarm interactive learning (IIL) behavior is triggered to adjust sizes of these two swarms.

In order to avoid the different individuals to repetitively exploit the same researched regions or excessive individuals to prematurely search a narrow region, researchers conducted a series of studies. Inspired by biology, niche [18] and speciation [19] techniques are introduced into PSO to prevent the swarm from crowding too closely and to locate as many optimal solutions as possible. Additionally, PSO topological structures are also widely adopted. The ring topology employed in [20] operating as a niching algorithm can drive the particles exploring the search space more broadly. Further, in [21] the multilayer strategy with multiple topologies is adopted to decrease the amount of noneffective exploiting tries. Inspired by ecological cohabitation, chaotic multiswarm particle swarm optimization (CMS-PSO) modifies the generic PSO with the help of the chaotic sequence for multidimension unknown parameter estimation and optimization by forming multiple cooperating swarms [22]. Historical memory strategy in HMPSO [23], which estimates and preserves distribution information of particles’ historical promising, is helpful in preserving the information of optimum solution space and making a comprehensive learning.

Although all the above works improve the performance of particle swarm optimization, there is shortage of research into the performance of the hybrid algorithm with the dynamic multiple swarms strategy and nonrepetition search approach.

In this paper, we propose a new coevolution particle swarm optimization algorithm with dynamic multispecies strategy based on K-means clustering and nonrepetition strategy based on Binary Space Partitioning fitness tree (called MCPSO-PSH). It is shown that when a Binary Space Partitioning tree archive stores the positions and the fitness values of all evaluated solutions, this archive can be treated as an approximation of the fitness function, which can avoid the individuals’ revisit phenomenon and improve the search efficiency. Moreover, K-means clustering method is introduced to partition whole population into subpopulations frequently. Information communication among subpopulations is implemented in the dynamic repartition process. Therefore, MCPSO-PSH algorithm can accommodate a considerable potential for solving complex problems. To comprehensively evaluate the performance of the proposed algorithm, MCPSO-PSH is compared with other state-of-the-art algorithms on three categories of experiments: (1) ten 10-dimensional and 30-dimensional benchmark problems with various properties, such as unimodal and multimodal functions, are employed to test MCPSO-PSO’s performance for a diverse set of problems; (2) a set of CEC2005 30-dimensional tests including 10 benchmark functions are used to justify MCPSO-PSH’s scalability for complex problems; (3) a real-world problem, the multilevel thresholds based on Otsu method for image segmentation, is employed to benchmark MCPSO-PSH’s applicability for practical problems. The numerical results demonstrate that MCPSO-PSH significantly improves the performance of PSO and outperforms most of the comparison algorithms during the experiments. The main contributions of the proposed methods lie in the following aspects.

(1) A new way of information communication among multiple swarms is designed. Each particle’s position is updated according to not only its personal history information and global information of its locating species but also other global information of other species.

(2) A dynamic K-means cluster is designed. In the process of decreasing the number of clusters and repartitioning the population into species, the search feature general achieves the transition from global search in the earlier evolution period to local search in the later evolution period.

(3) Binary Space Partitioning fitness tree memorizing the previous search history can prevent the individuals from revisiting the unpromising landscapes, which
can improve the search efficiency and avoid trapping in local optimum.

The rest of this paper is organized as follows: Section 2 first gives a review of the related works; Section 3 gives a brief explanation of the proposed coevolution particle swarm optimization algorithm; Section 4 presents the experimental studies of the proposed MCPSO-PSH; in Section 5, the performance of MCPSO-PSH based multilevel thresholding for image segmentation is evaluated. Finally, Section 6 concludes the paper.

2. Related Works

2.1. Particle Swarm Optimization. Particle swarm optimization (PSO) is an evolutionary computation (EC) algorithm paradigm that emulates the swarm behaviors of birds flocking [24]. It is a population-based search algorithm that exploits a population of individuals to probe promising regions of the search space. In this context, the population is called a swarm, and the individuals are called particles. Each particle moves with an adaptable velocity within the search space and retains the best position it ever encountered in its memory. The standard version of PSO is briefly described here [6].

Let $s$ be the swarm size, $d$ be the particle dimension space, and each particle of the swarm have a current position vector $X_i$, a current velocity vector $V_i$, and an individual best position vector $P_i$ found by the particle itself so far. The swarm also has the global best position vector $P^g$ found by any particle during all prior iterations in the search space. Assuming that the function $f(X)$ is to be minimized, and describing the following notations in $r$th iteration, the definitions are as follows:

$$X_i (t) = (x_{i,1} (t), x_{i,2} (t), \ldots, x_{i,d} (t)),$$

where each dimension of a particle is updated using the following equations:

$$V_{i,d} (t+1) = w \cdot V_{i,d} (t) + c_1 \cdot r_1 \cdot (P_{i,d} (t) - x_{i,d} (t)) + c_2 \cdot r_2 \cdot (P^g_{d} (t) - x_{i,d} (t)),$$

$$x_{i,d} (t+1) = x_{i,d} (t) + V_{i,d} (t+1).$$

In (2), $c_1$ and $c_2$ denote constant coefficients and $r_1$ and $r_2$ are elements from random sequences in the range of $(0, 1)$.

The inertia weight $w$ plays the role of balancing the global and local searches. It can be a positive constant (e.g., 0.9) or even a positive linear or nonlinear function of time [13, 14]. Although sometimes proper fine-tuning of the parameters $c_1$ and $c_2$ leads to an improved performance, an extended study of the cognitive and social parameters in [6] suggests $c_1 = c_2 = 2$ as default values.

2.2. K-Means Clustering. K-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. K-means clustering aims to partition $n$ observations into $k$ clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster, as shown in Figure 1. This results in a partitioning of the data space into Voronoi cells [25].

The cluster centers are substituted for center positions of food sources and the formula of computing the centers is shown in (4). If the $r$th cluster contains $n_r$ members and the members are denoted as $x^r_1, x^r_2, \ldots, x^r_{n_r}$, then the center (cluster $c^r_{center}$) is determined as

$$c^r_{center} = \frac{\sum_{i=1}^{n_r} x_i}{n_r}. \hspace{2cm} (4)$$

The radius $I$ of a cluster is defined as the mean distance (Euclidean) of the members from the center of the cluster. Thus, $R$ can be written as follows:

$$R_i = \frac{\sum_{p=1}^{n_i} \|x_p - c^r_{center}\|}{n_i}, \hspace{2cm} (5)$$

where $x_p$ is position of the $p$th member and $n_i$ is the number of members in its corresponding cluster.

2.3. Fitness Tree. Inspired by Binary Space Partitioning (BSP) tree, fitness tree is proposed by Shiu and Chi [26]. The establishment of a fitness tree is aimed at partitioning the whole search space into several subspaces (subregions) according to the history positions of members.

Each node of fitness tree represents a partitioned subspace of the whole search space. And each node has at most two child nodes, left node and right node. The sum of the search spaces of left node and right node is the search space of the parent node. Such fitness tree is constructed by a series of solutions found by the individuals. The process of building fitness tree is random, which means its topology for information communication is constructed stochastically.
3. Algorithm

3.1. Multispecies Coevolution Model for Optimization.

Inspired by the mutualism phenomenon, the single population PSO is extended into the interacting multispecies model by enhancing particle dynamics. In such coevolution model, the information for the particle's position updating includes not only its personal history information and global information of its locating species but also other global information of other species. Information communication simultaneously occurs in and between the local species and neighboring species.

In mathematical terms, the original particle's fly velocity equation and position update equation (please see (2) and (3)) are improved as follows:

\[ \begin{align*}
\mathbf{v}^k_{i,d}(t+1) &= \mathbf{w} \cdot \mathbf{v}^k_{i,d}(t) + c_1 \cdot r_1 \cdot (P_{i,d}(t) - x^k_{i,d}(t)) \\
&\quad + c_2 \cdot r_2 \cdot (P_{g,d}^k(t) - x^k_{i,d}(t)) + c_3 \cdot r_3 \\
\mathbf{x}^k_{i,d}(t+1) &= \mathbf{x}^k_{i,d}(t) + \mathbf{v}^k_{i,d}(t+1),
\end{align*} \]

where the superscript \( k \) represents the \( k \)th species, \( x^k_{i,d} \) denotes the position of the \( i \)th particle of the \( k \)th species, \( P_{i,d} \) is the current particle's history best position, \( P_{g,d}^k \) is the history best position found by its locating species, and \( P_{g,d}^k \) is the history best position found by the whole population. In (6), \( c_1, c_2, \) and \( c_3 \) are the constant coefficients. And in this paper, \( c_1, c_2, \) and \( c_3 \) are set to have the same value, \( c_1 = c_2 = c_3 = 1.6667; r_1, r_2, \) and \( r_3 \) are the random values in the range of \( (0, 1) \).

In (6), \( c_1 \cdot r_1 \cdot (P_{i,d}(t) - x^k_{i,d}(t)) \) takes into account the individual's own experiences, \( c_2 \cdot r_2 \cdot (P_{g,d}^k(t) - x^k_{i,d}(t)) \) means the internal communication within the \( k \)th species, and \( c_3 \cdot r_3 \cdot (P_{g,d}^k(t) - x^k_{i,d}(t)) \) indicates the symbiotic coevolution between dissimilar species.

3.2. Dynamically K-Means Cluster. In order to implement the multispecies coevolution model, the stochastically generated population should be partitioned into \( n \) subspecies. The widely adopted K-means cluster method (please see Section 2.2) is employed for achieving the population division. The main drawback of K-means cluster is that the effect of population division depends on the initial distribution of stochastically generated individuals and the selection of initial division points. A relatively effective means is to set a large number of clusters.

**Algorithm 1:** Pseudocode of fitness tree construction.

```plaintext
Input: Fitness tree and A new solution
Output: Fitness tree
% A new solution \( z \) is inserting into the fitness tree
(1) Initialization: Current node is the root node \( Curr\_node = \text{root}, RF = 0 \)
(2) If \( (Curr\_node \text{ has two child nodes } x \text{ and } y) \)
(3) Define the comparing dimension \( j \):
(4) \[ j = \arg \max_{k \in [1:D]} |x_j - y_j| \]
(5) If \( |x_j - z| \leq |x_j - y_j| \)
(6) \( Curr\_node = \text{child node } x; \)
(7) Else
(8) \( Curr\_node = \text{child node } y; \)
(9) End If and Goto Step (2)
(10) Else
(11) Insert a “virtual” child node to \( Curr\_node \), create a “real” child node that records \( z \) “under” the virtual child node.
(12) End If
```

The individuals of an evolution algorithm can be regarded as a sequence of solutions \( \text{sq} = \{s(1), s(2), \ldots, s(n)\} \), where \( n \) is the population size. In each cycle, \( n \) parent individuals generate \( n \) child individuals. The child individual is inserted into the fitness tree as a leaf node. So fitness tree stores all the search positions \( \text{SQ} = \{s_1, s_2, \ldots \} \). This means that there is no revisit for the moment. For normal evolution algorithms, parent individuals with better solutions withstand higher selection pressure, which results in higher probability to revisit the search space. In the extreme case, an individual trapping into a local optimum may be selected over and over, which causes that after several iterations all the individuals are generated from such individual. The phenomenon of revisit leads to a precipitous loss of population diversity.

The pseudocode of fitness tree construction is given in Algorithm 1. Here, a detailed explanation is given as follows. In the process of inserting a solution into the fitness tree, the current node \( Curr\_node \) represents the survey node. When the survey dimension of current node is equal to the dimension of inserted node \( Curr\_node = s(i) \), the \( i \)th solution \( s(i) \) in the sequence \( \text{sq} = \{s(1), s(2), \ldots \} \) is a revisit.

The core idea of fitness tree is that, under finding out the dimension with max difference, the new individual locates as a leaf node or partitions a parent node (subspace) into two binary nodes (sub-subspace).
As shown in Section 3.1. After several optimization iterations, the total population is repartitioned into \( K \) clusters (please see (5)); the left term (Dis) and the right term (clusters) change. The left term (Dis) changes. The right term (clusters) is not constant. To enhance the ability of local search and the individuals in small clusters may be redistributed into the relatively larger clusters when the number of the clusters changes.

To balance the exploration and exploitation, the value of \( T \) is not constant. To enhance the ability of local search especially in early stages of searching, \( T \) should be larger than \( T_{i+1} \). Therefore, in our proposed algorithm, \( T_j \) is calculated as follows:

\[
T_j = \frac{m - i + 1}{\sum_{j=1}^{m} j} \cdot T. \tag{8}
\]

Moreover, in the process of optimization, the centers of two or more clusters move too close or overlap each other. In order to avoid this phenomenon and to control every cluster to locally search a part of landscape, the relatively small cluster in two or more adjacent clusters decomposes and the individuals in such cluster redistribute into other adjacent clusters. Evaluation criterion is listed as follows:

\[
\text{Dis}_{\text{clusters}} - R_{\text{cluster1}} - R_{\text{cluster2}} < 0.2 \cdot \min (R_{\text{cluster1}}, R_{\text{cluster2}}), \tag{9}
\]

where Dis_{\text{clusters}} is the distance between two adjacent clusters and \( R_{\text{cluster1}} \) and \( R_{\text{cluster2}} \) are the radiuses of two adjacent clusters (please see (5)); the left term (Dis_{\text{clusters}} - R_{\text{cluster1}} - R_{\text{cluster2}}) represents the gap between two adjacent clusters.

### 3.3. Adaptive Normalization of Population Members

In our proposed algorithm, the number of clusters is predefined as \( C = \{c_1, c_2, \ldots, c_m\} \), where \( c_1 > c_2 > \cdots > c_m \). During optimization, each individual in every cluster operates independently as the multispecies coevolution model (please see Section 3.1). After several optimization iterations, the whole population is repartitioned into \( c_{i+1} \) clusters; \( c_i \) and \( c_{i+1} \) are orderly got from \( C = \{c_1, c_2, \ldots, c_m\} \). Here, \( T_1 + T_2 + \cdots + T_m = T \), where \( T \) is the total number of iterations. As \( c_{i+1} \) is smaller than \( c_i \), the individuals in small clusters may be redistributed into the relatively larger clusters when the number of the clusters changes.

In order to avoid the phenomenon and to control every cluster to locally search a part of landscape, the relatively small cluster in two or more adjacent clusters decomposes and the individuals in such cluster redistribute into other adjacent clusters. Evaluation criterion is listed as follows:

\[
\text{Dis}_{\text{clusters}} - R_{\text{cluster1}} - R_{\text{cluster2}} < 0.2 \cdot \min (R_{\text{cluster1}}, R_{\text{cluster2}}), \tag{9}
\]

where Dis_{\text{clusters}} is the distance between two adjacent clusters and \( R_{\text{cluster1}} \) and \( R_{\text{cluster2}} \) are the radiuses of two adjacent clusters (please see (5)); the left term (Dis_{\text{clusters}} - R_{\text{cluster1}} - R_{\text{cluster2}}) represents the gap between two adjacent clusters.

### 3.4. Algorithmic Framework

In this paper, a new multispecies coevolutionary particle swarm optimization based on previous search history is proposed. The K-means clustering and constructed fitness tree are employed to determine which dimension has the largest distinguishable difference. First, the extreme points of the new generated population \( S_{\text{new}} \) and constructed fitness tree \( S_{\text{tree}} \) are determined by identifying the minimum and maximum values \( z_{\text{new}}^\text{min} \) and \( z_{\text{new}}^\text{max} \) for each dimension \( j \) = 1, 2, \ldots, \( n \). Then, by constructing the extreme points \( z_{\text{min}} = (z_{\text{min}}^1, z_{\text{min}}^2, \ldots, z_{\text{min}}^n) \) and \( z_{\text{max}} = (z_{\text{max}}^1, z_{\text{max}}^2, \ldots, z_{\text{max}}^n) \). Each dimension value of nodes in fitness tree \( s_{\text{node}} \) is then translated by subtracting the corresponding dimension value of extreme points and dividing the distribution range of corresponding dimension as follows:

\[
z_n(s_{i,j}) = \frac{s_{\text{node},j} - z_{\text{min}}}{z_{\text{max}} - z_{\text{min}}}. \tag{10}
\]

The pseudocode of the proposed algorithm is listed in Algorithm 2.

### 3.4. Algorithmic Framework

In this paper, a new multispecies coevolutionary particle swarm optimization based on previous search history is proposed. The K-means clustering applied to dynamically partition the whole population into multiple species. Fitness tree is employed to avoid that the individuals revisit the previous search regions.

The pseudocode of the proposed algorithm is listed in Algorithm 3.

---

**Algorithm 1**: Pseudocode of the proposed multispecies particle swarm optimization.

**Input**: Fitness tree and new generated individuals

**Output**: Fitness tree

(1) All nodes of Fitness tree \( S_{\text{tree}} = \{s_{\text{node1}}, s_{\text{node2}}, \ldots\} \) and new generated individuals \( S_{\text{new}} = \{s_1, s_2, \ldots\} \)

(2) \( S = S_{\text{tree}} \cup S_{\text{new}} \)

(3) For \( j = 1 \)

(4) Compute minimum extreme point of each dimension:

\[
z_{\text{new}}^\text{min} = \min (s_{\text{node}j})
\]

\[
z_j^\text{min} = \min (s_{\text{node}j}, z_{\text{new}}^\text{min})
\]

(5) Compute maximum extreme point of each dimension:

\[
z_{\text{new}}^\text{max} = \max (s_{\text{node}j})
\]

\[
z_j^\text{max} = \max (s_{\text{node}j}, z_{\text{new}}^\text{max})
\]

(6) For \( j = 1 \)

(7) Normalize each dimension of each individual:

\[
z_n(s_{i,j}) = \frac{s_{\text{node},j} - z_{\text{node},j}}{z_{\text{node},j} - z_{\text{node},j}}
\]

(8) End For

(9) End For

---

**Algorithm 2**: Pseudocode of adaptive normalization of population members.

**Input**: Fitness tree and new generated individuals

**Output**: Fitness tree

(1) All nodes of Fitness tree \( S_{\text{tree}} = \{s_{\text{node1}}, s_{\text{node2}}, \ldots\} \) and new generated individuals \( S_{\text{new}} = \{s_1, s_2, \ldots\} \)

(2) \( S = S_{\text{tree}} \cup S_{\text{new}} \)

(3) For \( j = 1 \)

(4) Compute minimum extreme point of each dimension:

\[
z_{\text{new}}^\text{min} = \min (s_{\text{node}j})
\]

\[
z_j^\text{min} = \min (s_{\text{node}j}, z_{\text{new}}^\text{min})
\]

(5) Compute maximum extreme point of each dimension:

\[
z_{\text{new}}^\text{max} = \max (s_{\text{node}j})
\]

\[
z_j^\text{max} = \max (s_{\text{node}j}, z_{\text{new}}^\text{max})
\]

(6) For \( j = 1 \)

(7) Normalize each dimension of each individual:

\[
z_n(s_{i,j}) = \frac{s_{\text{node},j} - z_{\text{node},j}}{z_{\text{node},j} - z_{\text{node},j}}
\]

(8) End For

(9) End For
Initialization:

1. Stochastically generate a population of \( N \) individuals in search space;
2. Set the numbers of clusters \( C = \{c_1, c_2, \ldots, c_m\} \) and the total number of iteration \( T \);
3. Calculate fitness values of individuals;
4. Normalize the dimensions of each individual as Algorithm 2;
5. Initialize the fitness tree as Algorithm 1;
6. \( t = 1; I = 1; \)

Loop:

7. \( \text{While} (t < T) \)
8. If \( (t > \sum t_i) \)
9. Partition the whole population into \( c_i \) multiple species based on \( K \)-means clustering; 
   \( I = I + 1; \)
10. End If
11. Update the position of individuals according to Eq. (6) and (7);
12. Normalize the dimensions of each individual as Algorithm 2;
13. Insert new individuals into fitness tree as Algorithm 1;
14. Update some individuals from fitness tree;
15. Calculate fitness values of individuals;
16. \( t = t + 1 \)
17. End

Algorithm 3: Pseudocode of multispecies coevolution particle swarm optimization based on previous search history.

4. Experimental Study

4.1. Experimental Settings. The proposed multispecies coevolution particle swarm optimization based on previous search history (MCPSO-PSH) was executed with four state-of-the-art EA and SI algorithms for comparisons: (1) canonical particle swarm optimization (PSO) [24]; (2) differential evolution (DE) [27]; (3) Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [28]; (4) nonrevisiting genetic algorithm (NrGA) [26].

In all experiments in this section, the values of the common parameters used in each algorithm such as population size and total generation number are chosen to be the same. Population size is set as 100 and the maximum evaluation number is set as 100000. For the continuous testing functions used in this paper, the dimensions are all set as 10D and 30D.

All the control parameters for the EA and SI algorithms are set to be default of their original literatures.

For canonical PSO, the learning rates \( c_1 \) and \( c_2 \) are both set as 2.0.

For canonical DE, the zoom factor is set as \( F = 0.9 \) and crossover rate is set as \( CR = 0.9 \).

For CMA-ES, the population size \( \lambda \) is chosen by the suggested setting in [28] (i.e., \( \lambda = 4 + [3 * \ln D] \)).

For NrGA, the crossover rate of 0.8 and the mutation rate of 0.01 are adopted [26].

For our proposed algorithm, the number of clusters is set as \( C = \{10, 5, 4, 2, 1\} \), and so \( T \) is calculated as \( T = \{T_1, T_2, T_3, T_4, T_5\} = \{333, 267, 200, 133, 67\} \). The learning rates \( c_1, c_2, \) and \( c_3 \) in (6) are all set as \( c_1 = c_2 = c_3 = 1.3667 \).

4.2. Test and Benchmark Problems. According to the No Free Lunch theorem, "for any algorithm, any elevated performance over one class of problems is exactly paid for in performance over another class" [29]. To fully evaluate the performance of the MCPSO-PSH without a biased conclusion towards some chosen problems, a set of ten basic benchmark functions and ten CEC2005 benchmark functions [30] is employed. The formulas of these functions are presented below.

4.2.1. Basic Benchmark Functions.

1. Rosenbrock function

\[
f_1(x) = \sum_{i=1}^{n} 100 \times (x_{i+1}^2 - x_i^2)^2 + (1 - x_i)^2.
\]

2. Quadric function

\[
f_2(x) = \sum_{i=1}^{n} \left( \sum_{j=1}^{i} x_j \right)^2.
\]

3. Sum of different powers

\[
f_3(x) = \sum_{i=1}^{n} |x_i|^{i+1}.
\]

4. Axis parallel hyperellipsoid function

\[
f_4(x) = \sum_{i=1}^{n} c_i x_i^2.
\]

5. Sphere function

\[
f_5(x) = \sum_{i=1}^{n} x_i^2.
\]
Ackley’s function

\[ f_6 (x) = -20 \exp \left( -0.2 \sqrt{ \frac{1}{n} \sum_{i=1}^{n} x_i^2 } \right) \]
\[ - \exp \left( \frac{1}{n} \sum_{i=1}^{n} \cos 2\pi x_i \right) + 20 + e. \]

(16)

Griewank’s function

\[ f_7 (x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{i}} \right) + 1. \]

(17)

Michalewicz function

\[ f_8 (x) = -\sum_{i=1}^{n} \sin (x_i) \cdot \left( \sin \left( \frac{i \cdot x_i^2}{\pi} \right) \right)^{2m}, \quad m = 10. \]

(18)

Rastrigin’s function

\[ f_9 (x) = \sum_{i=1}^{n} x_i^2 - 10 \cos (2\pi x_i) + 10. \]

(19)

Schwefel’s function

\[ f_{10} (x) = \sum_{i=1}^{n} A_i x_i - B_i + f_{\text{bias}_{10}}. \]

(20)

These ten benchmark functions can be grouped as unimodal continuous functions \( f_1 \sim f_5 \) and multimodal continuous functions \( f_6 \sim f_{10} \).

4.2.2. CEC2005 Benchmark Functions. (11) Shifted sphere function

\[ f_{11} (x) = \frac{D}{1000} \sum_{i=1}^{D} z_i^2 + f_{\text{bias}_{11}}, \]
\[ z = x - o, \quad x = [x_1, x_2, \ldots, x_D]. \]

(21)

(12) Shifted Schwefel’s problem 1.2

\[ f_{12} (x) = \sum_{i=1}^{D} \left( \sum_{j=1}^{i} z_j \right)^2 + f_{\text{bias}_{12}}, \]
\[ z = x - o, \quad x = [x_1, x_2, \ldots, x_D]. \]

(22)

(13) Shifted rotated high conditioned elliptic function

\[ f_{13} (x) = \sum_{i=1}^{D} \left( \frac{10}{D} \right)^{i-1/D-1} z_i^2 + f_{\text{bias}_{13}}, \]
\[ z = (x - o) \cdot M, \quad x = [x_1, x_2, \ldots, x_D]. \]

(23)

(14) Shifted Schwefel’s problem 1.2 with noise in fitness

\[ f_{14} (x) = \sum_{i=1}^{D} \left( 100 (z_i^2 - z_{i+1})^2 + (z_i - 1)^2 \right) + f_{\text{bias}_{14}}, \]
\[ z = x - o + 1, \quad x = [x_1, x_2, \ldots, x_D]. \]

(24)

(15) Schwefel’s Problem 2.6 on bounds

\[ f_{15} (x) = \max \left( |A_i x - B_i| \right) + f_{\text{bias}_{15}}, \]
\[ i = 1, \ldots, D, \quad x = [x_1, x_2, \ldots, x_D]. \]

(25)

(16) Shifted Rosenbrock’s function

\[ f_{16} (x) = \sum_{i=1}^{D} \left( \sum_{j=1}^{i} z_j \right)^2 \cdot (1 + 0.4 |N(0, 1)|) + f_{\text{bias}_{16}}, \]
\[ z = x - o, \quad x = [x_1, x_2, \ldots, x_D]. \]

(26)

(17) Shifted rotated Griewank’s function without bounds

\[ f_{17} (x) = \sum_{i=1}^{D} \frac{z_i^2}{1000} - \prod_{i=1}^{D} \cos \left( \frac{z_i}{\sqrt{i}} \right) + 1 + f_{\text{bias}_{17}}, \]
\[ z = (x - o) \cdot M, \quad x = [x_1, x_2, \ldots, x_D]. \]

(27)

(18) Shifted rotated Ackley’s function on bounds

\[ f_{18} (x) = -20 \exp \left( -0.2 \sqrt{ \frac{1}{D} \sum_{i=1}^{D} z_i^2 } \right) \]
\[ - \exp \left( \frac{1}{D} \sum_{i=1}^{D} \cos (2\pi z_i) \right) + 20 + e + f_{\text{bias}_{18}}, \quad z = (x - o) \cdot M. \]

(28)

(19) Shifted Rastrigin’s function

\[ f_{19} (x) = \sum_{i=1}^{D} (z_i^2 - 10 \cos (2\pi z_i) + 10) + f_{\text{bias}_{19}}, \]
\[ z = (x - o) \cdot M, \quad x = [x_1, x_2, \ldots, x_D]. \]

(29)

(20) Schwefel’s Problem 2.13
unimodal functions
MCPSO-PSH, NrGA, and CMA-ES. For the low-dimensional and DE algorithms demonstrate a worse performance than
Table 1. From Table 1, it is obvious that the canonical PSO and DE algorithms demonstrate a worse performance than MCPSO-PSH, NrGA, and CMA-ES. For the low-dimensional (10D) unimodal functions $f_1$-$f_5$, the uniformity of CMA-ES algorithm is relatively good. For function $f_1$, CMA-ES gets the best results. And, for functions $f_2$-$f_5$, the performance of CMA-ES is rather competitive. Compared with the canonical
PSO and other state-of-the-art algorithms, MCPSO-PSH brings in the dynamical multispecies strategy based on K-means to enhance the ability of local search and fitness tree to overcome the revisit phenomenon, which results in a huge improvement of performance. The MCPSO-PSH and NrGA algorithms with nonrevisit strategy have nearly the same results for low-dimensional problems. Benefitting from the dynamic K-means clustering and the adaptive normalization of population members, the improved multispecies PSO has a slightly better performance than NrGA on functions $f_1$-$f_5$. For the low-dimensional (10D) multimodal functions $f_6$-$f_{10}$, intuitively, the proposed MCPSO-PSH algorithm has rather smaller mean values and standard deviations than other algorithms for function $f_6$. For functions $f_7$ and $f_8$, the MCPSO-PSH and NrGA obtain the global optimum with a high efficiency. It is due to the fact that the nonrevisit strategy based on BSP fitness tree effectively enhances the global search especially when a large number of local optima exist in the search space. For functions $f_9$ and $f_{10}$, the CMA-ES algorithm also gets the global optimum because the MCPSO-PSH, NrGA, and CMA-ES own their special mechanisms for jumping out of the local optimum. On the whole, regardless

$$f_{20}\left(x\right) = \sum_{i=1}^{D}\left(A_i - B_i\left(x\right)\right) + f_{\text{bias}_{20}},$$

$$x = \left[x_1, x_2, \ldots, x_D\right], A_i = \sum_{j=1}^{D}\left(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j\right), B_i = \sum_{j=1}^{D}\left(a_{ij}\sin x_j + b_{ij}\cos x_j\right).$$

These ten benchmark functions can be grouped as unimodal continuous functions $f_{11}$-$f_{15}$ and multimodal continuous functions $f_{16}$-$f_{20}$.

### 4.3. Result and Statistical Analysis

#### 4.3.1. Results for Basic Benchmark Functions

In this experiment, the proposed MCPSO-PSH algorithm is compared with nonrevisiting genetic algorithm (NrGA), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), the canonical particle swarm optimization (PSO) algorithm, and differential evolution (DE) algorithm.

For testing 10D benchmark test functions $f_1$-$f_{10}$ in 10 runs, the basic statistical results including the mean and standard deviations of the function values are listed in Table 1. From Table 1, it is obvious that the canonical PSO and DE algorithms demonstrate a worse performance than MCPSO-PSH, NrGA, and CMA-ES. For the low-dimensional (10D) unimodal functions $f_1$-$f_5$, the uniformity of CMA-ES algorithm is relatively good. For function $f_1$, CMA-ES gets the best results. And, for functions $f_2$-$f_5$, the performance of CMA-ES is rather competitive. Compared with the canonical

<table>
<thead>
<tr>
<th>Problem</th>
<th>MCPSO-PSH</th>
<th>NrGA</th>
<th>CMA-ES</th>
<th>PSO</th>
<th>DE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>Mean</td>
<td>4.313E-02</td>
<td>3.940E-01</td>
<td>4.957E-05</td>
<td>7.383E+04</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>7.378E-07</td>
<td>3.106E-05</td>
<td>1.473E-05</td>
<td>6.348E+03</td>
</tr>
<tr>
<td>$f_2$</td>
<td>Mean</td>
<td>8.037E-33</td>
<td>3.772E-27</td>
<td>8.499E-18</td>
<td>1.038E-03</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>1.037E-30</td>
<td>7.055E-30</td>
<td>3.873E-21</td>
<td>2.780E-03</td>
</tr>
<tr>
<td>$f_3$</td>
<td>Mean</td>
<td>9.493E-38</td>
<td>5.770E-33</td>
<td>3.495E-14</td>
<td>1.430E+01</td>
</tr>
<tr>
<td>$f_4$</td>
<td>Mean</td>
<td>2.904E-73</td>
<td>3.884E-36</td>
<td>8.399E-18</td>
<td>9.403E+08</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>4.789E-99</td>
<td>1.389E-45</td>
<td>2.939E-20</td>
<td>1.037E+09</td>
</tr>
<tr>
<td>$f_5$</td>
<td>Mean</td>
<td>4.955E-64</td>
<td>3.909E-39</td>
<td>4.139E-43</td>
<td>8.004E-17</td>
</tr>
<tr>
<td>$f_6$</td>
<td>Mean</td>
<td>2.213E-14</td>
<td>7.367E-13</td>
<td>3.893E-13</td>
<td>7.349E+01</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>3.789E-17</td>
<td>2.465E-14</td>
<td>3.904E-14</td>
<td>3.749E-03</td>
</tr>
<tr>
<td>$f_7$</td>
<td>Mean</td>
<td>0.000000</td>
<td>0.000000</td>
<td>7.432E-03</td>
<td>3.905E+00</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>4.780E-03</td>
<td>9.480E-03</td>
<td>9.480E-03</td>
<td>8.478E-03</td>
</tr>
<tr>
<td>$f_8$</td>
<td>Mean</td>
<td>-9.843E-01</td>
<td>-9.843E-01</td>
<td>9.057E-01</td>
<td>2.077E-01</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>0.000000</td>
<td>0.000000</td>
<td>6.904E-01</td>
<td>1.343E-01</td>
</tr>
<tr>
<td>$f_9$</td>
<td>Mean</td>
<td>0.000000</td>
<td>0.000000</td>
<td>3.894E+02</td>
<td>1.964E+03</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>0.000000</td>
<td>0.000000</td>
<td>9.037E+01</td>
<td>8.990E+01</td>
</tr>
<tr>
<td>$f_{10}$</td>
<td>Mean</td>
<td>0.000000</td>
<td>0.000000</td>
<td>1.335E+04</td>
<td>7.914E+04</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>0.000000</td>
<td>0.000000</td>
<td>8.944E+03</td>
<td>4.470E+04</td>
</tr>
</tbody>
</table>
of the unimodal functions or the multimodal functions, the performance of the proposed MCPSO-PSH is uniform.

For testing 3D benchmark test functions $f_1$ to $f_{10}$ in 10 runs, the basic statistical results including the mean and standard deviations of the function values are listed in Table 2. With the increase of the dimensions, the convergence rate becomes slow. The results of Table 2 show great disparities in convergence accuracy. For the high-dimensional unimodal problems $f_1$ to $f_5$, the performance of MCPSO-PSH algorithm is more outstanding than the other four algorithms because MCPSO-PSH with dynamic clustering strategy can accelerate the convergence rate especially in the later evolution period. The performance of NrGA algorithm becomes slightly poor although it also has the nonrevisit strategy for improving the search efficiency. Only for functions $f_1$ to $f_5$ does CMA-ES outperform NrGA on convergence rate and accuracy. For solving the high-dimensional multimodal problems $f_6$ to $f_{10}$, the MCPSO-PSH has a good performance. With the dynamic multispecies strategy based on K-means clustering and non-revisit strategy based on BSP fitness tree, the MCPSO-PSH algorithm can well avoid premature convergence to local optimum in the process of solving the high-dimensional multimodal problems. From the basic statistical results, NrGA also owns the most outstanding performance on testing functions $f_6$ and $f_{10}$, which means the nonrevisit strategy based on BSP fitness tree plays an important role in guiding search direction. On the whole, the performance of all algorithms tested here is ordered as MCPSO-PSH > NrGA > CMA-ES > PSO > DE.

The convergence curves of the MCPSO-PSH algorithms and other EAs against test functions $f_1$ to $f_{10}$ in 1D and 3D are shown in Figure 2. It can be observed that, with the increase of the dimensions, the convergence rate of the proposed MCPSO-PSH algorithm becomes faster. Predictably, with a further increase of the dimensions of the test functions, the MCPSO-PSH algorithm will have an overwhelming superiority in solving the high-dimensional problems.

### Table 2: Performance of all algorithms on 30D test functions $f_1$ to $f_{10}$

<table>
<thead>
<tr>
<th>Problem</th>
<th>MCPSO-PSH</th>
<th>NrGA</th>
<th>CMA-ES</th>
<th>PSO</th>
<th>DE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>Mean</td>
<td>4.8910E-01</td>
<td>1.3047E+01</td>
<td>3.1794E+00</td>
<td>3.4903E+12</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>3.3890E-01</td>
<td>9.0360E+00</td>
<td>2.0935E+00</td>
<td>8.0410E+11</td>
</tr>
<tr>
<td>$f_2$</td>
<td>Mean</td>
<td>1.6851E-14</td>
<td>1.8440E-08</td>
<td>5.3882E+02</td>
<td>3.7840E+04</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>9.0376E-16</td>
<td>9.5040E-10</td>
<td>3.0318E+02</td>
<td>9.9011E+03</td>
</tr>
<tr>
<td>$f_3$</td>
<td>Mean</td>
<td>5.9990E-37</td>
<td>6.3890E+01</td>
<td>1.4890E-12</td>
<td>3.8964E+30</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>9.5490E-41</td>
<td>1.4378E+01</td>
<td>2.1809E-13</td>
<td>1.9430E+31</td>
</tr>
<tr>
<td>$f_4$</td>
<td>Mean</td>
<td>1.0099E-16</td>
<td>8.3701E-08</td>
<td>1.0454E-23</td>
<td>7.9016E+13</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>8.4098E-18</td>
<td>1.0871E-07</td>
<td>9.5487E-25</td>
<td>1.0346E+14</td>
</tr>
<tr>
<td>$f_5$</td>
<td>Mean</td>
<td>1.3933E-61</td>
<td>2.0031E-11</td>
<td>2.9890E-16</td>
<td>2.3498E+02</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>9.0809E-75</td>
<td>9.6036E-13</td>
<td>7.3892E-23</td>
<td>7.3954E+01</td>
</tr>
<tr>
<td>$f_6$</td>
<td>Mean</td>
<td>1.3343E-14</td>
<td>1.3408E-12</td>
<td>6.3378E-06</td>
<td>8.4723E+02</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>6.6066E-15</td>
<td>8.0651E-14</td>
<td>1.6999E-06</td>
<td>3.1704E+02</td>
</tr>
<tr>
<td>$f_7$</td>
<td>Mean</td>
<td>0</td>
<td>0</td>
<td>3.0966E-02</td>
<td>2.0330E-01</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>0</td>
<td>0</td>
<td>7.8012E-03</td>
<td>3.8918E-01</td>
</tr>
<tr>
<td>$f_8$</td>
<td>Mean</td>
<td>-2.8743E+01</td>
<td>-2.7945E+01</td>
<td>1.7604E+01</td>
<td>6.2307E-01</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>1.664E+00</td>
<td>2.0874E+00</td>
<td>7.3370E+00</td>
<td>3.1700E+14</td>
</tr>
<tr>
<td>$f_9$</td>
<td>Mean</td>
<td>3.3880E-12</td>
<td>8.5801E+00</td>
<td>2.1388E+00</td>
<td>4.3880E+04</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>7.3038E-13</td>
<td>5.8817E+00</td>
<td>2.9863E+00</td>
<td>8.3909E+03</td>
</tr>
<tr>
<td>$f_{10}$</td>
<td>Mean</td>
<td>7.9063E-01</td>
<td>3.6421E-04</td>
<td>2.4578E+03</td>
<td>2.3841E+04</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>4.1939E-01</td>
<td>1.3940E-04</td>
<td>2.7309E+03</td>
<td>1.7301E+04</td>
</tr>
</tbody>
</table>
Figure 2: Continued.

(a) Rosenbrock-10D

(b) Quadric-10D

(c) Sum-10D

(a') Rosenbrock-30D

(b') Quadric-30D

(c') Sum-30D
Figure 2: Continued.
Figure 2: Continued.
nonrevisiting strategy is implemented in both NrGA and MCPSO-PSH, the robustness of NrGA is still undesirable. MCPSO-PSH with dynamic multispecies strategy has great enhancement of exploring ability.

From the above observations, the superiority of MCPSO-PSH over other peer algorithms can be easily presumed.

5. Multilevel Threshold for Image Segmentation

5.1. Otsu Method. The Otsu multithreshold measure [31] proposed by Otsu has been popularly employed in determining whether the optimal threshold method can provide image
Figure 3: Box plots of results obtained by all algorithms. Here, 1 to 5 in the horizontal axis are indices of MCPSO-PSH, NrGA, CMA-ES, PSO, and DE, respectively (the box has lines at the lower quartile, median, and upper quartile values. The whiskers extend to the most extreme data points not considered outliers. Outliers (denoted by +) are data with values beyond 100 units of interquartile range).
segmentation with satisfactory results. The traditional Otsu method can be described as follows.

Let the grey levels of a given image with \( N \) pixels range over \([0, L - 1]\) and \( h(i) \) denote the number of the \( i \)th grey level pixels. \( P(i) \) is the probability of \( i \).

\[
N = \sum_{i=0}^{L-1} h(i), \\
P(i) = \frac{h(i)}{N},
\]

for \( 0 \leq i \leq L - 1. \)

Otsu defined the between-class variance as the sum of sigma functions of each class by (22)

\[
f(t) = \delta_0 + \delta_1,
\]

\[
\delta_0 = w_0 \cdot (u_0 - u_T)^2,
\]

\[
\delta_1 = w_1 \cdot (u_1 - u_T)^2,
\]

\[
\delta_1 = w_1 \cdot (u_1 - u_T)^2,
\]

where \( u_T \) is the mean intensity of the original image. In bilevel thresholding, the mean level of each class, \( (u_i) \), can be calculated as follows:

\[
u_0 = \sum_{i=0}^{t-1} i \times P_i / w_0,
\]

\[
u_1 = \sum_{i=t}^{L-1} i \times P_i / w_0,
\]

where

\[
w_0 = \sum_{i=0}^{t-1} P_i,
\]

\[
w_1 = \sum_{i=t}^{L-1} P_i.
\]

The optimal threshold can be derived by maximizing the between-class variance function

\[
t^* = \arg \max \left( f(t) \right).
\]

Bilevel thresholding based on the between-class variance can be extended to multilevel thresholding. The extended between-class variance function is calculated as follows:

\[
f(t) = \sum_{i=0}^{M} \delta_i,
\]

where \( M \) is the number of thresholds. And

\[
\delta_0 = w_0 \cdot (u_0 - u_T)^2,
\]

\[
\delta_j = w_j \cdot (u_j - u_T)^2,
\]

\[
\delta_M = w_M \cdot (u_M - u_T)^2,
\]

\[
u_0 = \sum_{i=0}^{t-1} i \times P_i / w_0,
\]

\[
u_j = \sum_{i=t}^{t+j-1} i \times P_i / w_j,
\]

\[
u_M = \sum_{i=t}^{L-1} i \times P_i / w_j.
\]

The optimal thresholds are found by maximizing the between-class variance function:

\[
[t_1, t_2, \ldots, t_M] = \arg \max \{ f([t_1, t_2, \ldots, t_M]) \}. \tag{38}
\]

Equation (38) is used as the objective function which is to be optimized (maximized). A close look into this equation will show that it is very similar to the expression for uniformity measure.

5.2. Experiment Setup. For evaluating the performance of the proposed algorithm (MCPSO-PSH), we implemented it on a wide variety of images. These images include the popular tested images used in previous studies [32, 33], named avion.ppm, lena.ppm, and hunter.pgm (available in [34]) and 41033.jpg, 62096.jpg, and 145086.jpg provided by the Berkeley Segmentation Database (BSDS) (available in [35]). The sizes of avion.ppm, lena.ppm, and hunter.pgm are 512 \( \times \) 512 and the sizes of 41033.jpg, 62096.jpg, and 145086.jpg are 321 \( \times \) 481. Each image has a unique grey-level histogram. Most of the images are difficult to segment due to multimodality of the histograms.

The performance of the proposed algorithm is demonstrated by comparing with those of other popular algorithms developed in the literature so far, for example, canonical particle swarm optimization (PSO), differential evolution (DE), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), and nonrevisiting genetic algorithm (NrGA). A comparison between the proposed algorithm and other methods is evaluated based on Otsu, which means (38) is regarded as fitness function to evaluate all involved algorithms. The parameters settings are identical to those introduced in Section 4.1.

The numbers of thresholds\( M \) investigated in the experiments were 2, 3, 4, 5, 7, and 9, while all the experiments were repeated 30 times for each image. The population size is 100 and the maximum number of FEs is 2000. The numbers of thresholds \( M - 1 = 2, 3, 4 \) can be realized as low-dimensional problems, which also can be solved through the traditional search method although suffering from the long CPU time. The numbers of thresholds \( M - 1 = 5, 7, 9 \) are regarded
as medium levels for image segmentation, which means the high-dimensional problem for image segmentation cannot be solved by the traditional search method.

In all experiments, to ensure that the initial values of each random algorithm are equal, we used the MATLAB command \(\text{rand}(X,Y)\) * (max \(L\) − min \(L\)) + min \(L\). max \(L\) and min \(L\) are the maximum and minimum greys of the tested images. Figure 4 presents six original images and their histograms.

5.3. Experimental Results of Multilevel Threshold

Case I: Multilevel Threshold Results with \(M−1 = 2, 3, 4\). We employ (38) as the fitness function to provide a comparison of performances. Since the classical Otsu method is an exhaustive searching algorithm, we regarded its results as the “standard.” Table 4 shows the fitness function values (with \(M−1 = 2, 3, 4\)) attained by Otsu methods. In Table 4, the mean computation time and corresponding optimal thresholds are listed. For \(M−1 > 4\), owing to the unbearable consumption of CPU time, we do not list the correlative values in our experiment. These “standard” results will be used to compare with the results attained by optimization methods.

It is well known that the EA-based Otsu method for multilevel thresholding segmentation can only accelerate the computation velocity. The attained results will just get incredibly close to the optimal results but fail to improve the optimal results. Using the proposed MCPSO-PSH and other evolution algorithms, the mean fitness function values and their standard deviations obtained by the test algorithms (with \(M−1 = 2, 3, 4\)) are shown in Table 5. The bigger mean values and smaller standard deviation listed in Table 5 mean that the algorithm gets the better results.

According to the mean CPU times shown in Table 6, there is no obvious difference between the EA-based Otsu methods. All the EA-based Otsu methods are suitably efficient and practical in terms of time complexity for high-dimensional image segmentation problems. Therefore, in comparison with the traditional Otsu algorithm, all the EA-based Otsu methods effectively shorten the computation time.

Among the EA-based Otsu methods tested listed in Table 5, MCPSO-PSH obtains results equal to or close to that obtained in the traditional methods when \(M−1 = 2, 3, 4\). Moreover, the standard deviations obtained by MCPSO-PSH are considerably small among the results obtained by the EA-based Otsu methods. This is attributed to the fact that the MCPSO-PSH has a good balance between the global and local search abilities. Therefore, the MCPSO-PSH-based Otsu method is suitable for the low-dimensional segmentation problems (\(M−1 = 2, 3, 4\)).

Case II: Multilevel Threshold Results with \(M−1 = 5, 7, 9\).

To further investigate the search performance of each population-based method on high-dimensional segmentation problems, we conduct these algorithms on image segment with \(M−1 = 5, 7, 9\). Due to the unacceptable computation time for the traditional Otsu method, it does not need to be assessed in this test. The average fitness value and standard deviation obtained by each of the EA-based Otsu methods are listed in Table 7, where the correlative results with the larger values and smaller standard deviations indicate the better achievements.

From Table 7, it is obvious that, with the growth in dimension, there is statistically significant gap among experiments using these EA-based Otsu methods, in terms of both efficiency (fitness values) and stability (standard deviation). Depending on the nonrevisit strategy based on Binary Space Partitioning fitness tree and dynamic multispecies strategy based on \(K\)-means clustering, MCPSO-PSH owns the best performance and stability on a high dimension, which is more efficient than the canonical PSO and other classical EAs for image segmentation with Otsu method. Moreover, NrGA, which used nonrevisit strategy, also gets more acceptable results than the other algorithms, except MCPSO-PSH. With the increase of dimensions, the fitness values and standard deviation demonstrate that the performance of CMA-ES-based Otsu method has greatly improved. The results shown in Table 7 prove that the MCPSO-PSH-based Otsu method is more suitable for resolving multilevel image segmentation problems.

As mentioned above, the EA-based Otsu methods adopt different strategies to conduct global and local searches. An elaborate optimizer should have a better balance between exploration and exploitation search abilities. The above tests show that the MCPSO-PSH-based Otsu method outperforms the other EA-based Otsu method for multilevel image segmentation, especially on high-dimensional thresholding (with \(M−1 = 5, 7, 9\)). Moreover, compared with the traditional Otsu method, this method shortens the computational time.
Figure 4: Continued.
6. Conclusions

In this paper, we have presented a new coevolution particle swarm optimization algorithm with dynamic multispecies strategy based on K-means clustering and nonrevisit strategy based on Binary Space Partitioning fitness tree (called MCPSO-PSH). With the help of storing and analyzing the positions and the fitness values of all evaluated solutions in Binary Space Partitioning tree archive, MCPSO-PSH can prevent its individuals from revisiting the same researched regions and improve the search efficiency. The proposed algorithm adopts K-means clustering method to
Table 5: Objective value and standard deviation by the compared EA-based methods on Otsu algorithm.

<table>
<thead>
<tr>
<th>Image</th>
<th>M</th>
<th>Objective values (standard deviation)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCPSO-PSH</td>
<td>NrGA</td>
</tr>
<tr>
<td>Avion</td>
<td>2</td>
<td>3.4413E4</td>
</tr>
<tr>
<td></td>
<td>0.1390</td>
<td>7.6695E - 12</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3.4489E4</td>
</tr>
<tr>
<td></td>
<td>0.2708</td>
<td>1.9569E - 2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3.4534E4</td>
</tr>
<tr>
<td></td>
<td>0.5044</td>
<td>0</td>
</tr>
<tr>
<td>Lena</td>
<td>2</td>
<td>1.7597E4</td>
</tr>
<tr>
<td></td>
<td>3.8348E - 12</td>
<td>3.8348E - 12</td>
</tr>
<tr>
<td></td>
<td>1.7764E4</td>
<td>1.7764E4</td>
</tr>
<tr>
<td></td>
<td>6.3873E - 3</td>
<td>3.8347E - 1</td>
</tr>
<tr>
<td></td>
<td>1.7827E4</td>
<td>1.7827E4</td>
</tr>
<tr>
<td></td>
<td>0.4322</td>
<td>1.1927E - 4</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1.3909E - 2</td>
<td>1.9174E - 12</td>
</tr>
<tr>
<td></td>
<td>0.1261</td>
<td>0.1262</td>
</tr>
<tr>
<td>41033</td>
<td>2</td>
<td>1.2753E4</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1.2885E4</td>
<td>1.2885E4</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1.2945E4</td>
<td>1.2945E4</td>
</tr>
<tr>
<td></td>
<td>1.398E - 3</td>
<td>1.3245E - 3</td>
</tr>
<tr>
<td>62096</td>
<td>2</td>
<td>2.7469E4</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2.7585E4</td>
<td>2.7585E4</td>
</tr>
<tr>
<td></td>
<td>0.5133</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2.7631E4</td>
<td>2.7631E4</td>
</tr>
<tr>
<td></td>
<td>0.2255</td>
<td>15222E - 2</td>
</tr>
<tr>
<td>145086</td>
<td>2</td>
<td>1.5754E4</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1.5982E4</td>
<td>1.5982E4</td>
</tr>
<tr>
<td></td>
<td>2.333E - 2</td>
<td>1.9174E - 12</td>
</tr>
<tr>
<td></td>
<td>1.6085E4</td>
<td>1.6085E4</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1.9174E - 12</td>
</tr>
</tbody>
</table>

Table 6: The mean CPU time of the compared EA-based methods on Otsu algorithm.

<table>
<thead>
<tr>
<th>Dim</th>
<th>MCPSO-PSH</th>
<th>NrGA</th>
<th>CMA-ES</th>
<th>PSO</th>
<th>DE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 dim</td>
<td>0.3398</td>
<td>0.3088</td>
<td>0.2802</td>
<td>0.2553</td>
<td>0.2478</td>
</tr>
<tr>
<td>3 dim</td>
<td>0.3588</td>
<td>0.3193</td>
<td>0.2206</td>
<td>0.2497</td>
<td>0.3170</td>
</tr>
<tr>
<td>4 dim</td>
<td>0.3221</td>
<td>0.3897</td>
<td>0.2264</td>
<td>0.2520</td>
<td>0.3418</td>
</tr>
<tr>
<td>5 dim</td>
<td>0.3087</td>
<td>0.3903</td>
<td>0.1991</td>
<td>0.2355</td>
<td>0.3481</td>
</tr>
<tr>
<td>7 dim</td>
<td>0.3021</td>
<td>0.3818</td>
<td>0.2068</td>
<td>0.2311</td>
<td>0.3583</td>
</tr>
<tr>
<td>9 dim</td>
<td>0.3406</td>
<td>0.3949</td>
<td>0.1932</td>
<td>0.2308</td>
<td>0.3390</td>
</tr>
<tr>
<td>Image</td>
<td>Objective values (standard deviation)</td>
<td>Objective value and standard deviation by the compared EA-based methods on Otsu algorithm.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>MCPSO-PSH</td>
<td>NrGA</td>
<td>CMA-ES</td>
<td>PSO</td>
</tr>
<tr>
<td>Avion</td>
<td>5</td>
<td>3.4561E4</td>
<td>3.4561E4</td>
<td>3.4197E4</td>
<td>3.4547E4</td>
</tr>
<tr>
<td></td>
<td>2.2681E-2</td>
<td>6.6013E-2</td>
<td>2.2062</td>
<td>10.5393</td>
<td>1.2606</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>3.4585E4</td>
<td>3.4585E4</td>
<td>3.4222E4</td>
<td>3.4576E4</td>
</tr>
<tr>
<td></td>
<td>0.1227</td>
<td>0.4133</td>
<td>1.3292</td>
<td>7.2486</td>
<td>3.1975</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>3.4598E4</td>
<td>3.4597E4</td>
<td>3.4233E4</td>
<td>3.4580E4</td>
</tr>
<tr>
<td></td>
<td>0.2757</td>
<td>1.0760</td>
<td>1.1419</td>
<td>11.0739</td>
<td>2.1899</td>
</tr>
<tr>
<td>Lena</td>
<td>5</td>
<td>1.7853E4</td>
<td>1.7853E4</td>
<td>1.7598E4</td>
<td>1.7829E4</td>
</tr>
<tr>
<td></td>
<td>1.1762E-2</td>
<td>0.1222</td>
<td>1.3751</td>
<td>26.0899</td>
<td>1.9589</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>1.7885E4</td>
<td>1.7885E4</td>
<td>1.7626E4</td>
<td>1.7857E4</td>
</tr>
<tr>
<td></td>
<td>0.5759</td>
<td>0.1317</td>
<td>1.9112</td>
<td>23.0333</td>
<td>4.2468</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>1.7898E4</td>
<td>1.7898E4</td>
<td>1.7641E4</td>
<td>1.7882E4</td>
</tr>
<tr>
<td></td>
<td>0.7920</td>
<td>0.1250</td>
<td>2.3463</td>
<td>6.4354</td>
<td>3.2478</td>
</tr>
<tr>
<td></td>
<td>1.3509</td>
<td>0.9022</td>
<td>3.0022</td>
<td>46.7651</td>
<td>9.3539</td>
</tr>
<tr>
<td></td>
<td>0.132607</td>
<td>1.6085</td>
<td>3.6985</td>
<td>61.6729</td>
<td>3.6627</td>
</tr>
<tr>
<td></td>
<td>0.6610E2</td>
<td>0.8640</td>
<td>2.6400</td>
<td>24.2906</td>
<td>3.0252</td>
</tr>
<tr>
<td>4033</td>
<td>5</td>
<td>1.2974E4</td>
<td>1.2974E4</td>
<td>1.2967E4</td>
<td>1.2938E4</td>
</tr>
<tr>
<td></td>
<td>1.7191E-2</td>
<td>0.1151</td>
<td>2.5816</td>
<td>19.4046</td>
<td>1.2793</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>1.3006E4</td>
<td>1.3006E4</td>
<td>1.2995E4</td>
<td>1.2968E4</td>
</tr>
<tr>
<td></td>
<td>0.220026</td>
<td>0.1466</td>
<td>1.9642</td>
<td>12.1857</td>
<td>3.9443</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>1.3021E4</td>
<td>1.3020E4</td>
<td>1.3013E4</td>
<td>1.3004E4</td>
</tr>
<tr>
<td></td>
<td>0.883857</td>
<td>1.2077</td>
<td>2.4069</td>
<td>5.8058</td>
<td>2.4286</td>
</tr>
<tr>
<td>62096</td>
<td>5</td>
<td>2.7658E4</td>
<td>2.7658E4</td>
<td>2.7654E4</td>
<td>2.7650E4</td>
</tr>
<tr>
<td></td>
<td>3.5718E-2</td>
<td>0.2520</td>
<td>1.0799</td>
<td>5.7383</td>
<td>2.2228</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>2.7682E4</td>
<td>2.7682E4</td>
<td>2.7677E4</td>
<td>2.7669E4</td>
</tr>
<tr>
<td></td>
<td>0.1319</td>
<td>6.7905E-2</td>
<td>1.1261</td>
<td>5.792</td>
<td>2.1289</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>2.7695E4</td>
<td>2.7695E4</td>
<td>2.7689E4</td>
<td>2.7682E4</td>
</tr>
<tr>
<td></td>
<td>0.545201</td>
<td>0.4215</td>
<td>1.6069</td>
<td>9.1974</td>
<td>1.1693</td>
</tr>
<tr>
<td>145086</td>
<td>5</td>
<td>1.6143E4</td>
<td>1.6143E4</td>
<td>1.6134E4</td>
<td>1.6108E4</td>
</tr>
<tr>
<td></td>
<td>2.7021E-2</td>
<td>3.0505E-2</td>
<td>4.0455</td>
<td>43.2172</td>
<td>3.4949</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>1.6195E4</td>
<td>1.6195E4</td>
<td>1.6184E4</td>
<td>1.6160E4</td>
</tr>
<tr>
<td></td>
<td>0.1298</td>
<td>5.3333E-2</td>
<td>3.5967</td>
<td>23.9921</td>
<td>2.7644</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>1.6217E4</td>
<td>1.6217E4</td>
<td>1.6205E4</td>
<td>1.6192E4</td>
</tr>
<tr>
<td></td>
<td>0.4759</td>
<td>0.2663</td>
<td>2.7680</td>
<td>15.9752</td>
<td>2.4381</td>
</tr>
</tbody>
</table>

While this research is promising, there is a large margin of improvement for future research:

(1) The number of clusters $C$ and the repartitioned cycle $T$ are preset in this work. However, according to our preliminary experiments, different values of $C$ and $T$ will impact the results of several functions. $C$ and $T$ determine the range and rate of information exchange among species. For getting more suitable results of most test cases, we preset these two values. Therefore, how to adaptively adjust the values of $C$ and $T$ with regard to various functions is to be explored in the next stage.

(2) A nonrevisit strategy based on Binary Space Partitioning fitness tree is employed to partition the whole search space into several subregions according to
the history positions of members. According to our preliminary observation of particles’ trajectory, most of the global best positions are local under a parent node or a small patch of leaf nodes in the fitness tree, especially in the later evolution period. Most of the nodes in fitness tree have a lack of vitality. Therefore, in the future work, we may modify the static fitness tree into a dynamic one with a node age or a lifecycle feature.

(3) We would also like to introduce the proposed dynamic multiswarm and nonrevisit strategies to other swarm-based algorithms. At the same time, applying MCPSO-PSH to address more real-world complex problems will also be studied in the future work.

**Conflicts of Interest**

The authors declare that there are no conflicts of interest regarding the manuscript.

**Acknowledgments**

This research is partially supported by Liaoning Province Education Administration (L2015360, L2014473, and L2015372) and Tianjin Province Science and Technology Projects (16ZLZDZF00150) and National Natural Science Foundation of China (61603261, 61503373, 61602343, and 51607122).

**References**


http://decsai.ugr.es/cvg/dbimagenes/.
