Information Sciences 494 (2019) 233–246

Contents lists available at ScienceDirect

Information Sciences

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

Niching particle swarm optimization with equilibrium factor for multi-modal optimization

Yikai Li, Yongliang Chen, Jinghui Zhong*, Zhixing Huang

Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, School of Computer Science and Engineering, South China University of Technology, Guangzhou, China

A R T I C L E   I N F O

Article history:
Received 7 April 2018
Revised 20 January 2019
Accepted 21 January 2019
Available online 23 April 2019

Keywords:
Evolutionary computation
Multi-modal optimization
Particle swarm optimization
Niching technique

A B S T R A C T

Multi-modal optimization is an active research topic that has attracted increasing attention from evolutionary computation community. Particle swarm optimization (PSO) with niching technique is one of the most effective approaches for multi-modal optimization. However, in existing PSO with niching methods, the number of particles around different niches varies distinctly from each other, which makes it difficult for the algorithm to find high-quality solutions in all niches. To address this issue, this paper proposes a new niching PSO with equilibrium factor named E-SPSO. Different from the existing niching PSOs, the numbers of particles in different niches have been kept in balance in E-SPSO. The velocity of each particle is influenced by not only the personal best particle and the global best particle, but also an equilibrium factor (EF). By using the equilibrium factor to update the velocities of particles, the particles can be allocated uniformly among the niches. In this way, the computation resources can be assigned to the niches in a more balanced manner, so that the algorithm can gain more population diversity and find high-quality solutions in all niches. Experimental results on eleven benchmark problems show that the proposed mechanism not only increases the number of optima found, but also improves the search efficiency.

1. Introduction

Multi-modal optimization, which requires finding all optimal solutions in the search space, is an active research topic that has a number of real applications such as clustering problems [26], inversion of teleseismic waves [14], simulation-based design of power systems [9] and feeder network design problems [25]. EAs are population-based heuristic search algorithms which have been shown effective in solving complex optimization problems [5,33]. Since EAs contain a population of solutions, they have natural advantages in locating multiple optimal solutions in a single run. However, without specific design, traditional EAs are difficult to locate all global and local optima in a single run with high accuracy.

Niching method is one of the most effective methods to identify more solutions for multi-modal optimization problems (MMOPs). The key idea of the niching method is to maintain a high population diversity by using techniques such as fitness sharing [10], clustering [44], crowding [24], and restricted tournament selection [11]. Although the niching method has been proven effective in improving the capability of EAs to find more optimal solutions with high accuracy, it often requires much more computation cost, for the niching method usually slows down the convergence speed of the algorithms.

* Corresponding author.
E-mail address: jinghuizhong@scut.edu.cn (J. Zhong).

https://doi.org/10.1016/j.ins.2019.01.084
0020-0255/© 2019 Elsevier Inc. All rights reserved.

Particle Swarm Optimization (PSO) is a well-known EA variant. It makes use of individuals' experience and information of the population to guide the search. PSO has been proven to be effective and robust when faced with complicated optimization problems [47,48]. In recent years, niching PSOs are commonly used for solving MMOPs, and a number of niching PSOs have been proposed, such as VPSO [37] and nichePSO [3]. Niching PSOs search for the optima with the help of multiple stable subpopulations [19]. Due to its advantages, niching PSO has become one of the most popular approaches for MMOPs and has also been applied to a range of practical applications, such as clustering problems [26] and inversion teleseismic waves [14]. Famous niching PSOs include FERPSO [18], SPSO [27], Ring topology PSO [20], and etc. Furthermore, to enhance the capability of finding more accurate solutions, local search method is also integrated with niching PSO to improve the search efficacy [32].

However, existing niching PSOs, including FERPSO and SPSO, still encounter the problem of distinct disequilibrium of particle numbers among different niches. The numbers of particles in different niches tend to vary from each other during the searching process, especially when the number of optima is relatively large. In niching PSO, if the number of particles in a niche is too small, the niche may not be able to find highly accurate solutions due to the lack of diversity. Therefore, balancing the number of particles among niches is important for niching algorithms to find high accurate solutions in all niches, especially on complex MMOPs that contain many optimal solutions.

Keeping this in mind, this paper proposes an equilibrium factor (EF) to guide the search process. The key idea is to decrease the differences among the numbers of particles in different niches. EF encourages the worst particles in the largest niche to move towards the smallest niche. This mechanism can not only maintain the solution accuracy of the former niche, but also increase the probability of finding better solution in the latter niche. We embed EF inside SPSO with local search, forming a new niching PSO named E-SPSO. Different from the existing niching PSOs, the difference of particle numbers among different niches have been decreased in E-SPSO. By using EF to update the velocity of particles, the particles are distributed more evenly in the niches. The proposed E-SPSO has two main advantages. First, it increases the number of optima found in complex problems. By reducing the number of particles in the largest niche, the algorithm can drive more particles to explore other search space, which is useful to find more optimal solutions. Second, it improves the search efficiency. By increasing the number of particles in niches with less particles, the particles can be distributed uniformly among the niches. In this way, all niches can have enough diversity to avoid local stagnation and to search for highly accurate solutions, which is useful for improving the search efficiency.

The remainder of this paper is organized as follows. Section 2 gives a brief overview of basic PSO, and the related work on PSO for MMOPs. In Section 3, the proposed method is introduced. Section 4 describes the experimental studies. Finally, the conclusion part is given in Section 5.

2. Preliminaries

In this section, some preliminaries of our work are introduced. First, the definition of multi-modal optimization problem is presented. Then, a brief introduction to the particle swarm optimization (PSO) algorithm is given. Last, some related works are presented.

2.1. Multi-modal optimization problem (MMOP)

MMOP is a challenging optimization problem in evolutionary computation community. It requires identifying multiple solutions in a single run. However, locating multiple local optima and the global ones at the same time introduces extra difficulties, for it requires the algorithm to maintain the different optima found and improve the accuracy during the search process.

It should be noted that the MMOP is similar to the group- or cluster-consensus problem [39,40] (G/CCP). First, both problems can be solved by multi-agent system, where the agents will gradually converge to multiple states. Second, in both problems, each agent can communicate with other agents to update its state. For MMOP, the agents are implicitly driven by a fitness function. While for G/CCP, the agents are explicitly driven by a directed graph and the weights associated to the graph. However, the connection among the agents of the two problems are different. For G/CCP, the state of an agent can be directly obtained by linearly combining the states of nearby connected agents and the weights associated to them. While for MMOP, due to the randomness in reproduction operators, the relationships between agents are changing dynamically. Thus, the new state of an agent may be influenced by different agents in different time steps.

2.2. Particle swarm optimization

Over the past decades, a number of methods have been proposed to solve the MMOP. Among others, the PSO, which is proposed by Kennedy and Eberhart [7], has been proven to be excellent in solving MMOPs [23]. In PSO, each solution is represented by a particle, which searches solutions in a multi-dimension space by learning from its own experience and other particle's past experience. Its position X and velocity V are updated according to the following equations:

$$V_i^d(t+1) = w * V_i^d(t) + c_1 * rand1_i^d\left(pbest_i^d(t) - X_i^d(t)\right) + c_2 * rand2_i^d * \left(gbest^d(t) - X_i^d(t)\right) \tag{1}$$

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1) \tag{2}$$

where t represents the current generation, d refers to a dimension of the individual, and i is the index of a particle. c_1 and c_2 are the acceleration constants. $rand1_i^d$ and $rand2_i^d$ are two random numbers from a uniform distribution within the range of [0,1]. The inertia weight w is used to balance the global and local search performance. $pbest_i$ is the previously visited position yielding the best fitness value for the ith particle while *gbest* is the best position found by the whole population. In traditional PSO, Eq. (1) shows how the velocity of each particle will be updated during each iteration. *pbest* means the personal best position, which is the best position that a particle has ever identified. *gbest* means the global best particle in the whole population, which is the best position that the whole population has ever identified. X is the current position of a particle. w, $c1$, $c2$ are parameters that are used to control the velocity of a particle. During each iteration, the velocity of a particle is influenced by *pbest* and *gbest* on the basis of the particle's current velocity. And the particles tend to be attracted by *pbest* and *gbest*, as expressed in Eq. (1). With the help of *pbest* and *gbest*, the particles tend to converge around a global (or local) optimum. Eq. (2) shows how the position of a particle will be updated. In traditional PSO, the position of each particle is calculated by its current position and velocity from Eq. (1).

2.3. Related work on PSO for MMOP

PSO has been verified to have great potential in solving optimization problems and many enhanced PSO variants have been developed in the past decade. Existing PSOs for MMOP can be generally classified into two types. The first type focuses on preventing the particles from moving into the optima areas which have been identified during the searching process. To achieve this goal, it is important to improve the population diversity and different techniques have been proposed in the literature [8,42]. The second type focuses on the niching methods [19,32]. The key idea is to identify multiple optima using a group of niches and then maintain the identified optima till the end of the algorithm.

In the first type of algorithms, different methods have been proposed. For example, Parsopoulos and Vrahatis [28] proposed a PSO algorithm cooperated with some transformation techniques to solve this problem. In their work, the deflection and stretching techniques are used for transformation of the objective functions, and a repulsion source at each detected optima is used to ensure that a particle will be repelled away from moving towards the detected optima. Ran et al. [35] proposed a novel PSO called self-adaptive escape PSO, which uses a special mutation-escape operator to make particles explore the search space more efficiently. It produces a large speed value dynamically according to the variation of the speed, which encourages the particles to explore the local and global optima thoroughly at the same time. Michael et al. [31] proposed a PSO with Single Particle Repulsivity to solve the multi-modal optimization problem. They use a repulsive mechanism to improve the performance of the algorithm. Amit and Dilip [6] proposed a real-coded genetic algorithm (RCGA) by using a novel restart strategy. Lin et al. [22] proposed an improved differential evolution with searching pioneer in this field. They guided the vectors towards potential solution space by sharing parameter information of the vector that performed best in previous iteration with the other vectors. Li and Tan [16] proposed a loser-out tournament based fireworks algorithm, suggesting using competition as a new manner of interaction. In their new method, the fireworks are compared with each other according to both their current status and their progress rate. Antonia et al. [45] proposed a LaF-CMAES hybrid to achieve a balance between exploration and exploitation, in which techniques with distinct mechanisms for the task of exploration and the task of exploitation are developed.

In the second type of algorithms, niching techniques are used to improve the performance. For example, Brits et al. [3] proposed the NichePSO, which utilizes a group of subswarms to locate multiple optimal solutions. The subswarms are formed from an initial swarm by monitoring the fitness of individual particles. They can be merged together or absorb other individuals from the main swarm. The authors also proposed nbestPSO [4], in which a particle's neighborhood is defined by its $k-$closest particles and the particle's neighbourhood best is defined as the average position of its neighbors. Parrott and Li [27] proposed speciation-based PSO (SPSO). Li utilized the notion of species [17] and the size of species is determined by a radius parameter. The procedure for identifying species seeds, suggested by Pétrowski in [30] and Li et al. in [15] is also incorporated. The algorithm is then extended aiming to improve its robustness to the parameter r_s [1]. Bird and Li [1] proposed the Adaptive Niching PSO (ANPSO). It combats the problem of predefining niche radius by defining the radius as the average distance between a particle and its closet neighbor instead. Li [18] proposed the Fitness Euclidean-distance Ration Based PSO (FERPSO). He adopted the idea of Fitness-Euclidean-distance ratio (FER) from FDR-PSO (Fitness-Distance-Ratio based PSO) [43]. In FERPSO, a parameter named FER (Fitness and Eulicdean-distance Ratio) is calculated in order to determine a particle's fittest-and-closet neighborhood point p_n. The particles will be attracted towards p_n, instead of a single global best in canonical PSO. Schoeman and Engelbrecht [37] proposed vector-based (VPSO). They utilized the vector operations to determine the size of a niche and treated each individual as a vector. The process of determining the niche is terminated in a sequential manner and the algorithm doesn't require any predefined parameter. A parallel version of VPSO named PVPSO [36] is also proposed, which keeps all the niches in parallel and merges the niches when the distance between two niches is below a defined threshold. Passaro and Starita [29] proposed k-means clustering PSO (kPSO). Their work is inspired by Kenny's work [12], in which the k-means clustering algorithm is first used in a PSO algorithm to determine the personal bests and neighborhood bests. In kPSO, the parameter k is specified with the help of Bayesian information criterion (BIC) [38] and the k-means is applied to the swarm population at a regular interval. Li

[20] proposed ring topology PSO. It allows different neighborhood bests to coexist with the help of a ring topology and all the particles interact only with their immediate neighbors in this mechanism. The idea for searching with topology comes from [13], in which population structure is used to combine with PSO. In ring topology PSO, different niches will be formed around the identified neighborhood bests p_ns if the population size is large enough. All the niching methods have achieved some progress in MMOP and the related work is still going on in these years.

The PSO methods mentioned above hope to solve the MMOP by increasing the population diversity. With the help of searching in different subswarms, more optima can be identified and maintained till the end of the search. Nevertheless, all these methods have no mechanism to balance the distribution of particles. This paper proposed a general and effective balancing mechanism which can be applied to improve existing Niching PSOs.

3. The proposed method

The corresponding pseudocode is given in Algorithm 1. Generally, the proposed E-SPSO consists of four main steps,

Algorithm 1 The pseudocode of E-SPSO.

```
1:  Randomly generate an initial population, set the value of species radius $r$
2:  while Termination criterion is not met do
3:      $L_{sorted} \leftarrow$ all the particles sorted in decreasing order of fitness value
4:      Niche seeds $S \leftarrow \emptyset$
5:      while $L_{sorted} \neq \emptyset$ do
6:          $p \leftarrow$ the current fittest particle from $L_{sorted}$
7:          The seed of $p$ $seed_p \leftarrow \emptyset$
8:          for all the particles $s$ in $S$ do
9:              if distance$(p, s) < r$ then
10:                 $seed_p \leftarrow s$ ($p$ falls into the same niche with $s$)
11:                 break
12:             end if
13:         end for
14:         if $seed_p == \emptyset$ then
15:             $S \leftarrow S \bigcup \{p\}$ (Sort niche seeds in $S$ in decreasing order of fitness)
16:             $seed_p \leftarrow p$ ($p$ become the seed of a new niche)
17:         end if
18:         $L_{sorted} \leftarrow L_{sorted} - \{p\}$
19:     end while
20:     for $i = 1 \rightarrow populationsize$ do
21:         $nbest_i \leftarrow seed_i$
22:     end for
23:     Find the largest niche ($LN$) and the smallest niche ($SN$)
24:     Sort all particles in ($LN$) in decreasing order of fitness values
25:     $DV = SeedL - SeedS$, $DS = (sizeL - sizeS)/2$
26:     Update the velocities and positions for the last $DS$ particles in $LN$ with Eq. (9) and Eq. (10)
27:     Update the rest particles with Eq. (8) and Eq. (10)
28:     Update $pbest$ for every particle
29:     Local search
30: end while
```

namely initialization, construction of the niche, updating velocity and position, and local search procedure. It should be noted that the major contribution of this paper lies in Step 3, while the other three steps are based on the existing SPSO [27]. The meanings of important abbreviations in this paper are listed in Table 1. Fig. 1 compares the distribution of particles for E-SPSO and SPSOLS on F1, while Table 2 gives more details of the results in Fig. 1. The comparison results indicate that particles in E-SPSO are allocated more uniformly than those in SPSOLS.

Step-1 Initialization

Similar to traditional PSO algorithm [7], the first step in E-SPSO is to randomly generate NP (population size) particles to form an initial population. All the particles are uniformly distributed around the search space at first. The aims of the first step are to initialize the particles' positions and to initialize their velocities. Both position and velocity are represented in the form of vector. The dimension number D is specified according to the concrete problem. Specifically, the first step is executed based on the following equations:

$$X_i^d = rand_i^d(RangL_i^d, RangR_i^d) \tag{3}$$

$$Vmax^d = RangR^d - RangL^d \tag{4}$$

Table 1
Important abbreviations and their meanings.

Abbreviation	Meaning of abbreviation
ANFE	Average number of function evaluations
ANFO	Average number of optima found
ANOI	Average number of iterations
FERPSO	Fitness Euclidean-distance Ration Based PSO
FERPSOLS	Fitness Euclidean-distance Ration Based PSO with local search
SPSO	Speciation Based PSO
SPSOLS	Speciation Based PSO with local search
MMOP	Multi-modal optimization problems
SR	Success rate
EF	Equilibrium factor
E-SPSO\EF	E-SPSO without equilibrium factor
E-SPSO\LS	E-SPSO without local search

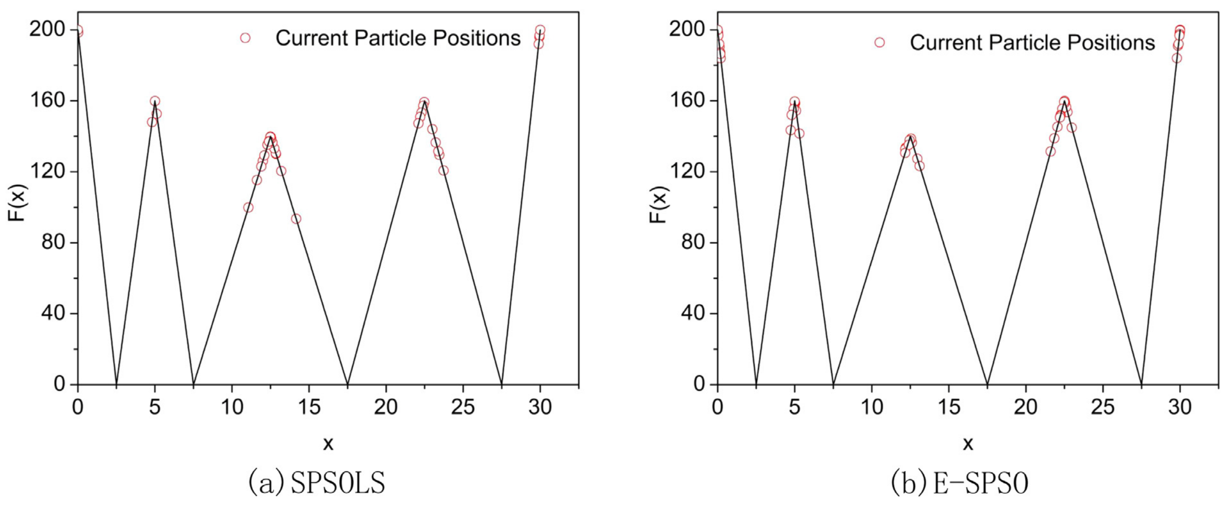

Fig. 1. Distribution of particles for SPSOLS and E-SPSO on F1. The curve of test problem F1 and the distribution of particles for SPSOLS and E-SPSO are shown. The figure shows that particles of E-SPSO are allocated more uniformly among the niches than SPSOLS.

Table 2
Distribution of particles for SPSOLS and E-SPSO on F1.

	Niche 1	Niche 2	Niche 3	Niche 4	Niche 5	Standard deviation	Variance	Range
SPSOLS	9	10	19	4	8	5.522681	30.5	15
E-SPSO	10	15	9	8	8	2.915476	8.5	7

$$V_i^d = rand_i^d((-0.5) * Vmax^d, 0.5 * Vmax^d) \tag{5}$$

where $i \in [1, NP]$ is the index of a particle and $d \in [1, D]$ refers to the dimension. X_i^d means the dth element of the ith particle's position, and similarly, V_i^d means the dth element of the ith particle's velocity. $rand_i^d(a, b)$ returns a random number within the range of $[a,b]$. As a particle may fly out of the search area if the velocity is too large, parameter $Vmax^d$ is used to determine the bounds of the dth element in a particle's velocity. $RangR^d$ and $RangL^d$ represent the upper bound and the lower bound of the dth dimension of the search space respectively.

Step-2 Construction of the species

In order to better solve MMOPs, we adopt niching technique to help identify and maintain multiple optima. Like SPSO [27], in E-SPSO, the niches are named species and the best particle in each species is called a seed. The species are created around the seeds and are responsible for converging around the optima. Species in E-SPSO will be updated in each generation, as described in the following parts.

In each generation, all the particles will be sorted in the descending order of their fitness values. Particles with the same fitness value will be sorted according to their original order in the whole population. After ordering all the particles, the algorithm checks the particles one by one to see whether they are suitable to be a seed. A species radius r is used in this procedure to determine the size of a species. The best particle in the whole population will be selected as the seed directly and be put into a set S for seeds. In each generation, S is defined to be empty at first and then the seeds will be added to S.

The other particles will be tested one by one in order by using the species radius *r*. If the distance between a particle and a seed is smaller than (or equal to) *r*, the particle is then considered to be a member of the same species with the seed. If the distances between a particle and all seeds in *S* are greater than *r*, then the particle will be considered to be a new seed and be put into *S*. After checking all the particles, every particle is assigned to a species. It should be noted that the *k* nearest neighbors method [41] can also be used to replace the species radius *r* when determining the size of the species. Performances of the k nearest neighbor method and species radius are similar according to our empirical study. For simplicity, we adopt species radius *r* in this paper.

Step-3 Update velocity and position with equilibrium factor

In E-SPSO, each particle is guided by some anchor particles during each iteration. $pbest_i$ and $nbest_i$ are two kinds of anchor particles used in E-SPSO. In this paper, the best position found by particle *i* is named the personal best of *i* ($pbest_i$), while the current best position of its species seed is called the neighbor best of *i* ($nbest_i$). The anchor particles are adopted in velocity equation, which is similar to velocity equation of PSO. The only difference is that *gbest* is replaced by $nbest_i$ in E-SPSO. Velocity equation of E-SPSO is shown in Eq. (8) and (9).

In most cases, numbers of particles in different species will vary gradually during the search procedure. In order to minimize the difference, we propose the equilibrium factor (EF) mechanism, where the worst few particles in the largest species will be thrown out and driven to some other species. A deviation vector (*DV*) is added to the velocity equation of these particles so that they will move in a direction to the species that has fewer individuals.

When implementing EF, species will be sorted in the descending order of their sizes. The largest species and the smallest one will be picked out to calculate *DV*:

$$DV^d = seedS^d - seedL^d \tag{6}$$

where *seedL* is the seed of the largest species and *seedS* refers to the seed of the smallest one.

Meanwhile, all the individuals in the largest species will be sorted according to their fitness values. A parameter named deviation size (*DS*) is calculated to determine the number of particles that will adopt EF. *DS* is calculated by Eq. (7):

$$DS = 0.5 * (sizeL + sizeS) \tag{7}$$

where *sizeL* and *sizeS* are the numbers of particles in the largest species and the smallest one respectively. The aim of equation (7) is to balance the number of particles in the largest niche and the smallest niche, by moving a number of particles in the largest niche towards the smallest niche. The value of *DS* is set to be the average number of *sizeL* and *sizeS* in order to balance the number of particles in the largest and smallest niches after each iteration. Then, the worst *DS* particles in the largest niche will be influenced by EF. The EF mechanism specifies *DV* during each iteration and determines which particles will be influenced by the vector at the same time. *DV* is then used to update particles' velocities.

After calculating the *DV*, velocities and positions of the particles are updated by:

$$V_i^d(t+1) = w * V_i^d(t) + c_1 * rand1_i^d * \left(pbest_i^d(t) - X_i^d(t)\right) + c_2 * rand2_i^d * \left(nbest^d(t) - X_i^d(t)\right) \tag{8}$$

$$V_i^d(t+1) = w * V_i^d(t) + c_1 * rand1_i^d * \left(pbest_i^d(t) - X_i^d(t)\right) + c_2 * rand2_i^d * \left(nbest^d(t) - X_i^d(t)\right) + DV \tag{9}$$

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1) \tag{10}$$

where Eq. (9) is used for the particles adopt the EF and Eq. (8) is used for the particles that are not influenced by EF. The worst *DS* particles that have been selected by EF in the largest niche will use Eq. (9) to update their velocities, while the other particles will adopt Eq. (8). The positions of all the particles are updated according to Eq. (10).

Step-4 Local search

In the fourth step, all the *pbest*s will be updated according to the local search procedure proposed in [32]. Each *pbest* will generate a new particle around itself and compete with it. If the new particle has a better fitness value, it will replace the original *pbest* to become a new *pbest*. More specifically, for each *pbest* in the population, a fine-tuning operator takes place according to the difference between the original *pbest* and another *pbest* that is the nearest one to it ($pbest_{nearest}$).

Finally, a repetition between step 2 and step 4 will be executed until the termination condition is met.

4. Experiment studies

4.1. Experimental settings

In this section, the proposed E-SPSO is competed against seven state-of-the-art algorithms (i.e., LIPS [34], LoICDE [2], Fast-LIPS [50], Fast-NCDE [50], triDEMO [46], FERPSOLS [18,32] and SPSOLS [27,32]) to test its performance. Eleven commonly used multimodal benchmark functions of different types [21] are used for testing. The characteristic of these functions are presented in Table 3. The details of other parameter settings are listed in Table 4. Note that a successful found of a certain peak requires that the distance between a particle and the peak is smaller than the tolerance value (accuracy level). What's more, the acceleration constants and the inertia weight of all the algorithms are set to be the same [32]: c_1=2.05, c_2=2.05 and w=0.729843788. In the experiment, each benchmark will be run 200 independent times.

Table 3
Benchmark test functions.

Name	Test function	Range	Peaks global/ local
F1 Five-uneven-peak Trap	$f_1(x) = \begin{cases} 80(2.5-x) & 0 \le x < 2.5 \\ 64(x-2.5) & 2.5 \le x < 5 \\ 64(7.5-x) & 5 \le x < 7.5 \\ 28(x-7.5) & 7.5 \le x < 12.5 \\ 28(17.5-x) & 12.5 \le x < 17.5 \\ 32(x-17.5) & 17.5 \le x < 22.5 \\ 32(27.5-x) & 22.5 \le x < 27.5 \\ 80(x-27.5) & 27.5 \le x \le 30 \end{cases}$	$0 \le x \le 30$	2/3
F2 Equal Maxima	$f_2(x) = sin^6(5\Pi x)$	$0 \le x \le 1$	5/0
F3 Uneven Decreasing Maxima	$f_3(x) = exp[-2log(2).(\frac{x-0.08}{0.854})^2] *sin^6(5\Pi(x^{3/4}-0.05))$	$0 \le x \le 1$	1/4
F4 Himmelblua's function	$f_4(x,y) = 200 - (x^2+y-11)^2 -(x+y^2-7)^2$	$-6 \le x, y \le 6$	4/0
F5 Six-Hump Camel Back	$f_5(x,y) = -4[(4-2.1x^2+\frac{x^4}{3})x^2 +xy+(-4+4y^2)y^2]$	$-1.9 \le x, y \le 1.9$	2/4
F6 Shubert function	$f_6(\vec{x}) = p = -\prod_{i=1}^{2}\sum_{j=1}^{5} jcos[(j+1)x_i+j]$	$-10.0 \le x1, x2 \le 10.0$	18/many
F7 Vincent 1 Dimension	$f_7(x) = sin(10log(x))$	$0.25 \le x1 \le 10.0$	6/0
F8 Vincent 2 Dimension	$f_8(\vec{x}) = \frac{1}{2}\sum_{i=1}^{2} sin(10log(x_i))$	$0.25 \le x1, x2 \le 10.0$	36/0
F9 Modified Rastrigin 2 Dimension	$f_9(\vec{x}) = -\sum_{i=1}^{2}(10+9cos(2\Pi * 5x_i))$	$0.0 \le x1, x2 \le 1.0$	25/0
F10 Modified Rastrigin 5 Dimension	$f_{10}(\vec{x}) = -\sum_{i=1}^{5}(10+9cos(2\Pi * 2x_i))$	$0.0 \le x1, x2, x3, x4, x5 \le 1.0$	32/0
F11 Modified Rastrigin 6 Dimension	$f_{11}(\vec{x}) = -\sum_{i=1}^{6}(10+9cos(2\Pi * 2x_i))$	$0.0 \le x1, x2, x3, x4, x5, x6 \le 1.0$	64/0

Table 4
Population size, number of function evaluations and level of accuracy of LIPS [34], LoICDE [2], Fast-NCDE [50], Fast-LIPS [50], triDEMO [49], FERPSOLS [32], SPSOLS [32]and E-SPSO.

Function No.	Population size	No. of function evaluations	Level of accuracy
F1	50	10000	0.0005
F2	50	20000	0.0000001
F3	50	20000	0.0000001
F4	50	20000	0.0000001
F5	50	20000	0.000001
F6	250	100000	0.05
F7	100	20000	0.0001
F8	250	200000	0.001
F9	250	100000	0.0000001
F10	2400	200000	0.05
F11	2000	400000	0.05

In this paper, we use the following three criteria to evaluate the performance of the algorithms: 1) Success rate (SR), which describes the probability of a certain algorithm finding all the peaks in an MMOP; 2) Average number of found optima (ANFO), which means the average number of peaks found over 200 runs in a given accuracy, showing the recall ability of the algorithm; 3) Average number of function evaluations (ANFE), which measures the average time cost for an algorithm to identify all the required optima, reflecting the speed of convergence. When the success rate is zero, the ANFE equals the predefined maximum function evaluation times.

4.2. Results and discussions

The Wilconxon signed-rank test was performed to assess the significance of differences between the algorithms in terms of SR, ANFO and ANFE. The results are shown in Table 5 to 7, where symbols $+$, $-$ and $\approx$ represent that the competitor is respectively significantly better than, worse than and similar to E-SPSO, respectively, according to the Wilcoxon signed-rank test at $\alpha = 0.05$.

The success rates (SR) for the seven comparison algorithms are compared with E-SPSO and the results are presented in Table 5. It can be observed from the results that E-SPSO performs better in most cases. On F5, the value of SR is better than that of LIPS, Fast-LIPS, Fast-NCDE, triDEMO, SPSOLS, and FERPSOLS. On F9, the value of SR has been improved by 1307.69%

Table 5
Comparison of success rate (SR).

Test function	LIPS [34]	LoICDE [2]	Fast-NCDE [50]	Fast-LIPS [50]	tri-DEMO [46]	FERPSOLS [18,32]	SPSOLS [27,32]	E-SPSO
F1	0.98−	0.975−	0.96−	**1** ≈	0.905−	0.09−	0.965−	**1**
F2	**1** ≈	**1** ≈	**1** ≈	**1** ≈	0.985−	**1** ≈	**1** ≈	**1**
F3	0.97−	0.965−	**1** ≈	0.98−	0.885−	0.67−	0.97−	**1**
F4	**1** ≈	**1** ≈	**1** ≈	0.84−	0.82−	0.83−	**1** ≈	**1**
F5	0.12−	**0.14** ≈	0.11−	0.13−	0.12−	0−	0.11−	**0.14**
F6	0.82+	0.78+	0.385−	0.84+	0.615+	**0.86**+	0.235−	0.42
F7	**0.98** ≈	0.965 ≈	0.96−	**0.98** ≈	0.97 ≈	0.965 ≈	0.945−	**0.98**
F8	0≈	0.26+	0 ≈	0 ≈	**0.42**+	0 ≈	0 ≈	0
F9	0.88−	0.865−	0.87−	0.9−	0.895−	0.065−	0.895−	**0.915**
F10	0.95−	0.935−	0.97 ≈	0.96−	0.94−	0.92−	0.975−	**0.985**
F11	0 ≈	0 ≈	0 ≈	0 ≈	0 ≈	0 ≈	0 ≈	0

Symbols +, − and ≈ represent that the competitor is respectively significantly better than, worse than and similar to E-SPSO according to the Wilcoxon signed-rank test at $\alpha = 0.05$.

Table 6
Comparison of average number of found optima (ANFO).

Test function	LIPS [34]	LoICDE [2]	Fast-NCDE [50]	Fast-LIPS [50]	tri-DEMO [46]	FERPSOLS [18,32]	SPSOLS [27,32]	E-SPSO
F1 Mean	4.96 ≈	4.86−	4.84−	**5** ≈	4.38−	2.27−	4.965 ≈	**5**
Std	0.12	0.729	0.629	0	0.7825	0.8606997	0.18424	0
F2 Mean	**5** ≈	**5** ≈	**5** ≈	**5** ≈	4.92 ≈	**5** ≈	**5** ≈	**5**
Std	0	0	0	0	0.24	0	0	0
F3 Mean	4.955 ≈	4.925 ≈	**5**≈	4.94 ≈	4.32 ≈	3.68−	4.97 ≈	**5**
Std	0.25	0.252	0	0.352	0.426	1.88557	0.24973	0
F4 Mean	**4** ≈	**4** ≈	**4** ≈	3.7−	3.26−	3.82−	**4** ≈	**4**
Std	0	0	0	0.232	0.2632	0.41041	0	0
F5 Mean	4.652−	4.705 ≈	4.54−	4.68−	4.25−	1.955−	4.635−	**4.715**
Std	0.356	0.625	0.405	0.352	0.482	0.2078243	0.68858	0.70445
F6 Mean	17.84+	17.52+	15.96−	**17.86**+	17.25+	17.84+	16.12−	16.975
Std	0.23	1.86	1.232	0.569	0.426	0.4186602	1.11075	1.04874
F7 Mean	5.975 ≈	5.97−	5.965−	5.975 ≈	5.96−	5.96−	5.925−	**5.98**
Std	0.156	0.156	0.172	0.213	0.232	0.2205521	0.22855	0.14035
F8 Mean	25.42−	28.86+	27.56+	26.62−	**29.85**+	19.825−	26.08−	27.04
Std	1.3236	1.658	0.43296	1.3656	1.5286	2.144146	1.51163	1.750922
F9 Mean	24.82−	24.28−	24.56−	24.86−	24.76−	22.73−	24.825−	**24.91**
Std	0.476	0.468	0.452	0.3256	0.425	1.362491	0.34983	0.30391
F10 Mean	31.36−	31.06−	31.76−	31.48−	31.425−	27.565−	31.975−	**31.985**
Std	1.626	1.64	1.527	1.6542	1.6782	3.29461	2.16953	2.42321
F11 Mean	47.74−	46.94−	47.82−	47.85−	47.72−	45.09−	46.66−	**48.105**
Std	23.25	58.62	46.24	57.82	3.2856	4.337739	3.412127	3.556292

Symbols +, − and ≈ represent that the competitor is respectively significantly better than, worse than and similar to E-SPSO according to the Wilcoxon signed-rank test at $\alpha = 0.05$.

than FERPSOLS, 2.23% than triDEMO and 2.23% than SPSOLS. Meanwhile, the value is also improved on F1, F3, F4, F7 and F10. On F6, the SR of E-SPSO is improved by 78.72% than SPSOLS and 9.09% than Fast-NCDE, but the other few algorithms perform better than E-SPSO on this function. It shows that the recall capabilities of the algorithms are different on different test functions. On F2, all the algorithms perform well in all runs.

Table 6 shows that E-SPSO performs better than or at least competitively to the other algorithms in most cases. The values of ANFO have been raised by 10.94% than triDEMO, 1.73% than SPSOLS, 3.85% than Fast-NCDE and 141.18% than FERPSOLS on F5, respectively. As for F1, F3, F7 and F9, the values are also improved. On F6, the value of ANFO is raised by 5.30% than SPSOLS and by 6.36% than Fast-NCDE. But the other algorithms perform better on this test function, which is the same as the result in success rates (SR). While on F2, all of the eight algorithms identify nearly all the optima in the whole runs.

For most of the test functions, E-SPSO identifies all the optima within the required accuracy with a relatively less number of function evaluations. As can be seen in Table 7, on F4, the value of ANFE is improved by 23.09% than triDEMO, 14.31% than Fast-NCDE, 20.03% than Fast-LIPS, 4.73% than SPSOLS and 13.84% than FERPSOLS, respectively. As for F2, F5, F9 and F10, the ANFE value is also improved on average.

Table 7
Comparison of average number of function evaluations (ANFE).

Test function	LIPS [34]	LoICDE [2]	Fast-NCDE [50]	Fast-LIPS [50]	tri-DEMO [46]	FERPSOLS [18,32]	SPSOLS [27,32]	E-SPSO
F1 Mean	4731−	4631.5−	3831+	4622.5−	7117−	**1150**+	4003.25 ≈	3942.25
Std	771.31	881.23	826.47	819.2	931.526	487.356	863.731	877.684
F2 Mean	11023−	9639.5−	11131−	11522.5−	12506−	16729−	9678.6−	**9421.5**
Std	1634.5	1923.26	2118.1	1623.3	1776.52	1703.4	1549.166	1471.665
F3 Mean	8826+	8126+	9126.5+	8912+	12031−	**640.909**+	9589.9−	9331
Std	1632.57	1391.07	1327.42	1314.16	319.932	261.085	1617.763	1343.302
F4 Mean	15734−	16170−	17621.5−	19120−	19632.5−	17525.3−	15850−	**15100**
Std	1837.5	2632.115	2298.263	3421.32	3617.46	1965.65	3733.965	3800.453
F5 Mean	18394.5−	17620−	16612.5−	16346−	17812.5−	–	18418.18−	**16315.385**
Std	2893.22	3771.345	2623.32	2596.46	3316.27	–	3630.512	2820.594
F6 Mean	33222.5+	27405+	28160+	**26630**+	31085+	41000+	27066.25+	51750
Std	1412.57	1062.25	1645	1630.22	1805.46	2306.215	1544.205	3718.35
F7 Mean	**10269**+	12652−	11972 ≈	11828 ≈	11529.5+	28122.28−	10772.63+	12097.96
Std	2169.24	2352.82	2652.46	1852.56	2252.62	8559.64	5280.486	2722.027
F8 Mean	–	122380	–	–	130000	–	–	–
Std	–	8037.65	–	–	6157.25	–	–	–
F9 Mean	111140−	149580−	114130−	111630−	110737.5−	576832.5−	101743.025−	**90750.25**
Std	10640.2	16910.21	20240.52	17630.65	18905.45	51014.75	15918.695	18093.73
F10 Mean	163707−	199658−	175662−	181632−	193130−	1187200−	62938.776−	**42076.584**
Std	13080.21	15650.46	16400.52	11050.16	10615.87	13268.46	5708.384	4965.88
F11 Mean	–	–	–	–	–	–	–	–
Std	–	–	–	–	–	–	–	–

Symbols +, − and ≈ represent that the competitor is respectively significantly better than, worse than and similar to E-SPSO according to the Wilcoxon signed-rank test at $\alpha = 0.05$.

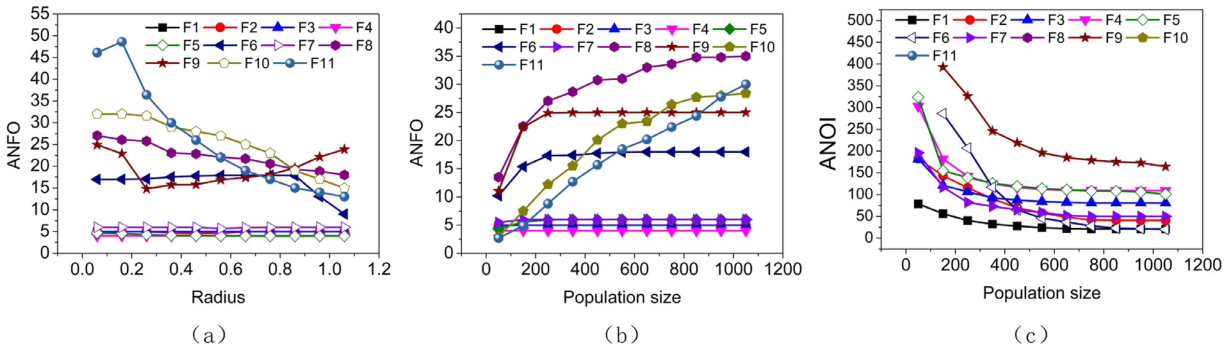

Fig. 2. The influence of species radius on ANFO, and the influence of population size on ANFO and ANOI.

4.3. Algorithm analysis

After comparing against the other seven algorithms, the proposed E-SPSO is tested on F1-F11 by changing the parameter species radius, population size and *DS* value. The results are evaluated by the following three criteria: average number of found optima (ANFO), average number of function evaluations (ANFE) and average number of iterations (ANOI). The experimental results shown in Fig. 2 and Fig. 3 demonstrate that the performance of E-SPSO is influenced by all the three parameters.

The values of average number of optima (ANFO)found are influenced by species radius, population size and *DS* value. On F5, E-SPSO performs the best when the value of species radius is set to 0.06. The value of ANFO decreases when species radius becomes larger. On F6, the value of ANFO increases at first. When radius grows larger, the value of ANFO decreases. The best value of species radius on F6 is about 0.86. On F9, the experimental results are opposite to the results on F6. The value of ANFO decreases at first. But it increases when species radius grows larger. Experimental results on species radius show that the best values of species raius are different for different test problems, but E-SPSO performs relatively well on most test functions when species raiuds is set to 0.06. The results on population size show that a larger population has a better recall capability in E-SPSO. When the value of population size increases, more optima can be found on average. The performance of E-SPSO is also influenced by the *DS* value. Experiment results show that when $w = 0.5$, E-SPSO performs well on most of the test problems.

For all the eleven test functions, the average number of iterations (ANOI) decreases when the population size grows larger. Experiment results show that a larger population helps E-SPSO speed up its convergence process. As for the influence

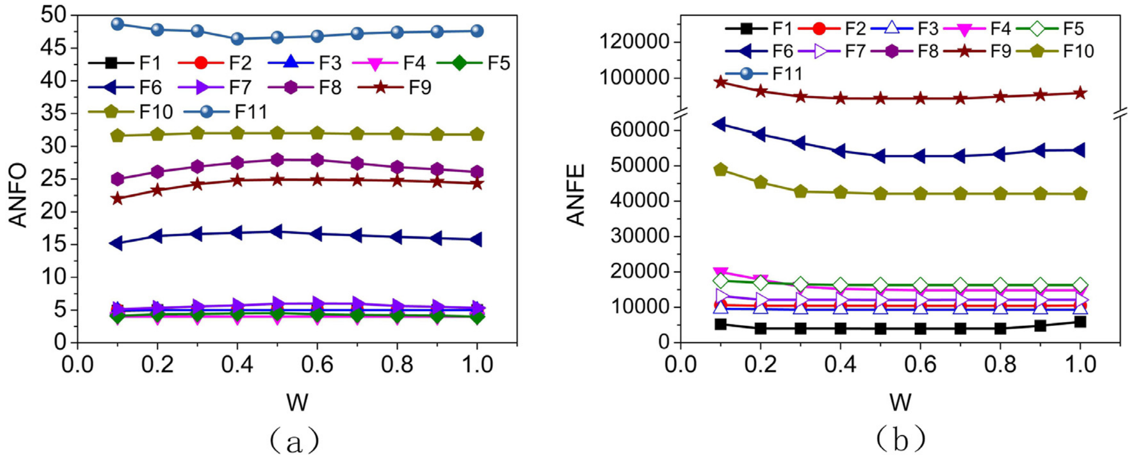

Fig. 3. The influence of parameter DS value on ANFO and ANFE. *W* is a parameter that is used to calculate the *DS* value, where $DS = W * sizeS + (1 - W) * sizeL$..

Table 8
Comparison of success rate (SR).

Function	E-SPSO/LS	E-SPSO/EF	E-SPSO
F1	0.925 – (–)	0.965–	**1**
F2	0.94 – (–)	**1** ≈	**1**
F3	0.87 – (–)	0.97–	**1**
F4	0.84 – (–)	**1** ≈	**1**
F5	0.12 - (≈)	0.11–	**0.14**
F6	0.24 - (≈)	0.235–	**0.42**
F7	0.895 – (–)	0.945–	**0.98**
F8	0 ≈(≈)	0 ≈	0
F9	0.785 – (–)	0.895–	**0.915**
F10	0.96 – (–)	0.975–	**0.985**
F11	0 ≈(≈)	0 ≈	0

Symbols – and ≈ outside brackets represent that the competitor is respectively significantly worse than and similar to E-SPSO according to the Wilcoxon signed-rank test at $\alpha = 0.05$. The symbols in brackets are the corresponding results between E-SPSO(without local search) and E-SPSO(without EF).

of *DS* value on average number of function evaluations (ANFE), results show that E-SPSO performs relatively better when $w = 0.5$ on most of the test problems.

Next, the influence of equilibrium factor and local search is investigated. Specifically, we conduct experiments on two simplified E-SPSO, named E-SPSO/EF (E-SPSO without equilibrium factor) and E-SPSO/LS (E-SPSO without local search). The success rates (SR) for E-SPSO, E-SPSO/EF and E-SPSO/LS are listed in Table 8. It can be observed that E-SPSO performs better in most cases. On F1, F3, F5, F6, F7, F9 and F10, E-SPSO achieves a higher success rate than the other two algorithms. As for F8 and F10, the SR values of the three algorithms are all zero. With the help of local search method and EF, E-SPSO is able to achieve a higher success rate.

Table 9 shows the results of ANFO for E-SPSO, E-SPSO/EF and E-SPSO/LS. On most of the test functions, E-SPSO identifies more optima on average. The value of ANFO has been improved by 5.30% than E-SPSO/EF and 14.39% than E-SPSO/LS on F6. On F2, all the algorithms manage to find all the optima in a single run.

In Table 10, when compared on the results of ANFE, the results show that E-SPSO performs better or competitively to E-SPSO/EF and E-SPSO/LS in most cases. On F5, the value has been improved by 11.42% than E-SPSO/EF and 13.73% than E-SPSO/LS. On F1, F2, F3, F4, F9, F10, E-SPSO also identifies all the optima with a smaller number of function evaluations. The above experimental results show that both local search method and EF help improve the searching efficiency.

Fig. 4 shows the influence of population diversity on E-SPSO. The diversity of population is determined by two aspects. One aspect is the species size, the other aspect is the number of species in the whole population. In Fig. 4(a), E-SPSO is tested on F1. The result shows that when species size grows larger, the algorithm will achieve a higher level of accuracy. In Fig. 4(b), E-SPSO is also tested on F1 to show how the number of species will influence the performance of E-SPSO. In E-SPSO, the swarm was divided into different species. Different species are responsible for searching different areas. If the number of species is too small, the algorithm may not be able to identify all the optima. Otherwise, if the number of species is large enough, the algorithm will be more likely to identify more optima. The curve in Fig. 4(b) shows that when the number of species is large enough, E-SPSO can identify more optima in a single run. Last, diversity curves of the eight

Table 9
Comparison of average number of found optima (ANFO).

Function	E-SPSO/LS	E-SPSO/EF	E-SPSO
F1 Mean	4.865 – (–)	4.965 ≈	**5**
Std	0.8606997	0.18424	0
F2 Mean	**5** ≈(≈)	**5** ≈	**5**
Std	0	0	0
F3 Mean	3.98 – (–)	4.97 ≈	**5**
Std	1.885571	0.24973	0
F4 Mean	3.72 – (–)	**4** ≈	**4**
Std	0.41041	0	0
F5 Mean	3.955 – (–)	4.635–	**4.715**
Std	0.2078243	0.68858	0.70445
F6 Mean	14.84 – (–)	16.12–	**16.975**
Std	0.4186602	1.11075	1.04874
F7 Mean	5.91 – (–)	5.925–	**5.98**
Std	0.2205521	0.22855	0.14035
F8 Mean	22.825 – (–)	26.08–	**27.04**
Std	2.144146	1.51163	1.750922
F9 Mean	23.73 – (–)	24.825–	**24.91**
Std	1.32491	0.34983	0.30391
F10 Mean	31.425 – (–)	31.975–	**31.985**
Std	2.426	2.16953	2.42321
F11 Mean	45.32 – (–)	46.66–	**48.105**
Std	3.526252	3.412127	3.556292

Symbols – and ≈ outside brackets represent that the competitor is respectively significantly worse than and similar to E-SPSO according to the Wilcoxon signed-rank test at $\alpha = 0.05$. The symbols in brackets are the corresponding results between E-SPSO(without local search) and E-SPSO(without EF).

Table 10
Comparison of average number of function evaluations (ANFE).

Function	E-SPSO/LS	E-SPSO/EF	E-SPSO
F1 Mean	4162.5 – (–)	4003.25 ≈	**3942.25**
Std	487.3565	863.731	877.684
F2 Mean	10121.24 – (–)	9678.6–	**9421.5**
Std	2084.4	1549.166	1471.665
F3 Mean	10176.25 – (–)	9589.9–	**9331**
Std	319.93225	1617.763	1343.302
F4 Mean	16283.5 – (–)	15850–	**15100**
Std	41835.705	3733.965	3800.453
F5 Mean	18912 – (–)	18418.18–	**16315.385**
Std	3346	3630.512	2820.594
F6 Mean	31132.5 + (–)	**27066.25**+	51750
Std	24162.725	1544.205	3718.35
F7 Mean	12635.45 – (–)	**10772.63**+	12097.96
Std	2202.3	5280.486	2722.027
F8 Mean	–	–	–
Std	–	–	–
F9 Mean	114400.5 – (–)	101743.025–	**90750.25**
Std	16374	15918.695	18093.73
F10 Mean	84299.088 – (–)	62938.776–	**42076.584**
Std	37630.2	5708.384	4965.88
F11 Mean	–	–	–
Std	–	–	–

Symbols +, – and ≈ outside brackets represent that the competitor is respectively significantly better than, worse than and similar to E-SPSO according to the Wilcoxon signed-rank test at $\alpha = 0.05$. The symbols in brackets are the corresponding results between E-SPSO(without local search) and E-SPSO(without EF).

algorithms are also compared in Fig. 5. The element-wise position diversity [49] is used to calculate the position diversity. The results show that E-SPSO maintains better position diversity during the process of iteration.

The reasons why E-SPSO can achieve promising performance are as follows. Firstly, the number of particles in the smallest niches is raised with the help of the equilibrium factor. As a consequence, searching capability in the smallest niche is improved at the same time. The phenomenon helps speed up the searching progress of the algorithm and makes it possible to identify more optima in a single run. Secondly, in the joint influence of the equilibrium factor and the original velocity equation, the particles added with the deviation vector are able to search in a larger scale of area. The equilibrium factor

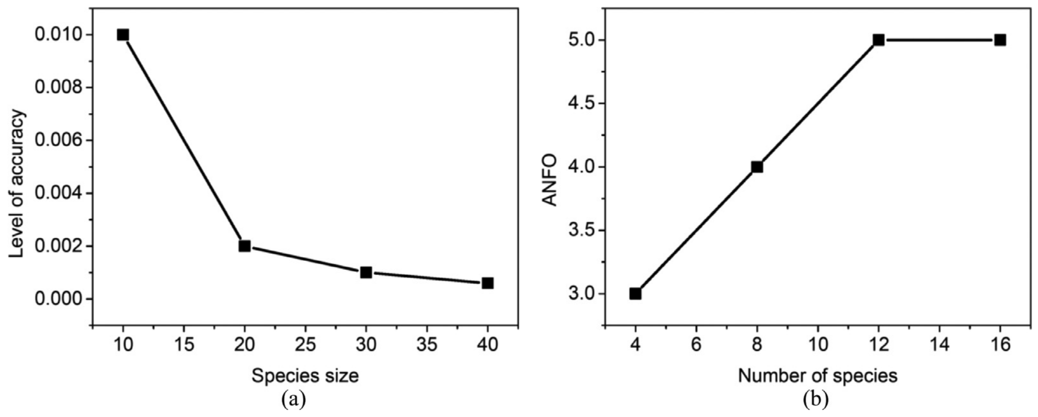

Fig. 4. The influence of species size on accuracy, number of species on ANFO.

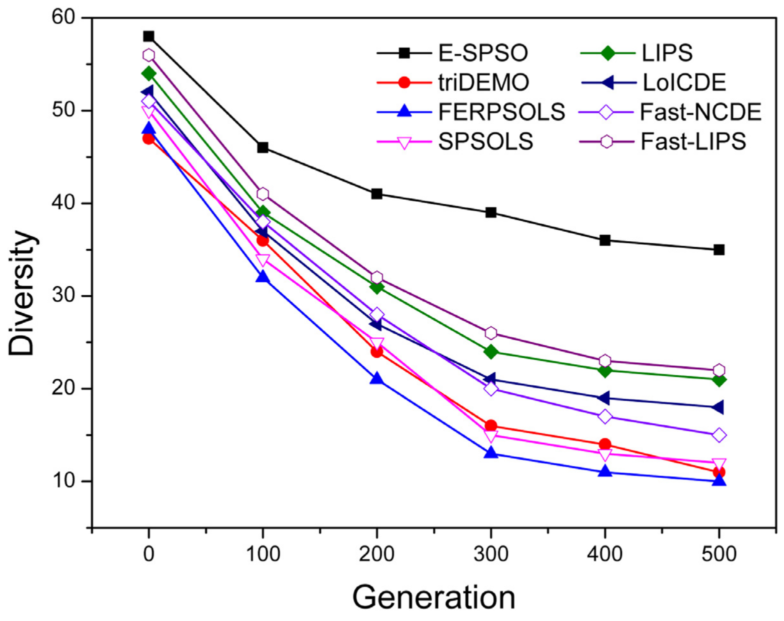

Fig. 5. Diversity curves on Five-uneven-peak Trap problem. The eight algorithms are tested on F1. And element-wise position diversity [49] is used to calculate the position diversity..

helps the chosen particles in the largest niche move towards the smallest niche by adding a deviation vector to the particles' velocity. And the velocity equation in SPSOLS helps all the particles in the population move towards their species seeds at the same time. Consequently, the chosen particles in the largest niche will move in the joint influence of the equilibrium factor and the original velocity equation. The velocity direction of the chosen particles are decided by the result of the deviation vector and the original velocity equation. Some of the chosen particles may move to other area expect the species seed and the smallest niche. The particles are able to search in a larger scale of area, with the help of the equilibrium factor and the velocity equation. As a result, more optima are able to be identified in the process and the success rate is raised as well.

5. Conclusion

In this paper, a diversity enhancement method named equilibrium factor mechanism, has been proposed to improve the performance of Niching PSO for multimodal optimization. The equilibrium factor not only can balance the distribution of particles in different niches, but also improves the probability of the algorithm to identify more optima in the searching space. The experimental results have shown that the proposed niching PSO with equilibrium factor outperforms the original niching PSO variants when faced with MMOPs.

There are several interesting future research topics. One is to improve the current migration strategy. In the proposed E-SPSO, the migration strategy for the particles is relatively simple. Only particles in the largest niches can be influenced by

EF and be moved towards the smallest niches during each iteration. Meanwhile, the number of particles that will be moved by EF is relatively fixed. In the future research work, the adaption strategy can be used in E-SPSO to improve its current migration strategy. The random number can also be used in the migration strategy to improve its performance. The second work is about parameter setting. In E-SPSO, the size of species is determined by species radius. In future, we plan to try some other mechanisms to determine the species size, for example, the *k* nearest neighbors method in which a topological distance is used to influence the size [41]. Another future work is applying the proposed equilibrium factor to improve other multi-modal EAs such as DE and GA. In addition, many problems in our daily life are MMOPs. The E-SPSO can be used to solve realistic problems in the future research work.

Conflict of interest

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled.

Acknowledgment

This work is partially supported under the National Natural Science Foundation of China (Grant No. 61602181), National Training Program of Innovation and Entrepreneurship for Undergraduates (No. 201810561127), and Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X183).

References

[1] S. Bird, X. Li, Adaptively choosing niching parameters in a PSO, in: Proceedings of the 8th annual conference on Genetic and evolutionary computation (GECCO '06), ACM, New York, USA, 2006, pp. 3–10.
[2] S. Biswas, S. Kundu, S. Das, Inducing niching behavior in differential evolution through local information sharing, IEEE Trans. Evol. Comput. 19 (2) (2015) 246–263.
[3] R. Brits, A.P. Engelbrecht, F.V.D. Bergh, A niching particle swarm optimizer, in: Conference on Simulated Evolution and Learning, 2002, pp. 692–696.
[4] R. Brits, A.P. Engelbrecht, F. Van den Bergh, Solving systems of unconstrained equations using particle swarm optimization, in: IEEE International Conference on Systems, Man and Cybernetics, 2003, p. 6pp. vol.3.
[5] W. Chu, X. Gao, S. Sorooshian, A new evolutionary search strategy for global optimization of high-dimensional problems, Inf. Sci. (Ny) 181 (22) (2011) 4909–4927.
[6] A.K. Das, D.K. Pratihar, A novel restart strategy for solving complex multi-modal optimization problems using real-coded genetic algorithm, in: International Conference on Intelligent Systems Design and Applications, 2017, pp. 32–41.
[7] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 1995, pp. 39–43.
[8] S.K.S. Fan, E. Zahara, A hybrid simplex search and particle swarm optimization for unconstrained optimization, Eur. J. Oper. Res. 181 (2) (2007) 527–548.
[9] A.Y. Goharrizi, R. Singh, A.M. Gole, S. Filizadeh, J.C. Muller, R.P. Jayasinghe, A parallel multimodal optimization algorithm for simulation-based design of power systems, IEEE Trans. Power Delivery 30 (5) (2015) 2128–2137.
[10] D.E. Goldberg, J. Richardson, Genetic algorithms with sharing for multimodal function optimization, in: International Conference on Genetic Algorithms on Genetic Algorithms and Their Application, 1987, pp. 41–49.
[11] G. Harik, Finding multimodal solutions using restricted tournament selection, 1995, pp. 24–31.
[12] J. Kennedy, Stereotyping: improving particle swarm performance with cluster analysis, in: Proceedings of the 2000 Congress on Evolutionary Computation, vol.2, 2000, pp. 1507–1512.
[13] J. Kennedy, R. Mendes, Population structure and particle swarm performance, in: Proceedings of the 2002 Congress on Evolutionary Computation, 2002, pp. 1671–1676.
[14] K.D. Koper, M.E. Wysession, D.A. Wiens, Multimodal function optimization with a niching genetic algorithm: a seismological example, Bull. Seismol. Soc. Am. 89 (4) (1999) 978–988.
[15] J.-P. Li, M.E. Balazs, G.T. Parks, P.J. Clarkson, A species conserving genetic algorithm for multimodal function optimization, Evol. Comput. 10 (3) (2002) 207–234, doi:10.1162/106365602760234081.
[16] J. Li, Y. Tan, Loser-out tournament based fireworks algorithm for multi-modal function optimization, IEEE Trans. Evol. Comput. PP (99) (2017). 1–1.
[17] X. Li, Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization, Lect. Notes Comput. Sci. 3102 (2004) 105–116.
[18] X. Li, A multimodal particle swarm optimizer based on fitness Euclidean-distance ratio, in: Proceedings of the 9th annual conference on Genetic and evolutionary computation (GECCO '07), ACM, New York, USA, 2007, pp. 78–85.
[19] X. Li, Developing niching algorithms in particle swarm optimization, in: Panigrahi B.K., Shi Y., Lim M.H. (Eds.), Handbook of Swarm Intelligence. Adaptation, Learning, and Optimization, vol 8, Springer, Berlin, Heidelberg, 2010.
[20] X. Li, Niching without niching parameters: particle swarm optimization using a ring topology, IEEE Trans. Evol. Comput. 14 (1) (2010) 150–169.
[21] X. Li, A. Engelbrecht, M.G. Epitropakis, Benchmark functions for cec'2013 special session and competition on niching methods for multimodal function optimization (2013) http://goanna.cs.rmit.edu.au/xiaodong/cec13-niching/competition/.
[22] C. Lin, S. Hsieh, S. Chiu, Improved differential evolution with searching pioneer for solving multi-modal optimization problems, in: 2017 Fifth International Symposium on Computing and Networking (CANDAR), 00, 2018, pp. 101–105, doi:10.1109/CANDAR.2017.92.
[23] S.W. Mahfoud, Niching methods for genetic algorithms, University of Illinois at Urbana-Champaign, Champaign, IL, USA, 1995 UMI Order No. GAX95-43663.
[24] R. Manner, S.W. Mahfoud, Crowding and preselection revisited, Parallel Parallel Problem Solving From Nature (1992) 27–36.
[25] A.S. Mohaymany, A. Gholami, Multimodal feeder network design problem: ant colony optimization approach, J. Eng. Mater. Technol. Trans. ASME 120 (1) (2010) 71–78.
[26] A. Naik, S.C. Satapathy, A.S. Ashour, N. Dey, Social group optimization for global optimization of multimodal functions and data clustering problems, Neural Comput. Appl. (2016) 1–17.
[27] D. Parrott, X. Li, Locating and tracking multiple dynamic optima by a particle swarm model using speciation, IEEE Trans. Evol. Comput. 10 (4) (2006) 440–458.
[28] K.E. Parsopoulos, M.N. Vrahatis, Modification of the Particle Swarm Optimizer for Locating all the Global Minima, Springer Vienna, 2001.

[29] A. Passaro, A. Starita, Particle swarm optimization for multimodal functions: a clustering approach, J. Artif. Evol. Appl. 2008 (2) (2008) 15.
[30] A. PéTrowski, A clearing procedure as a niching method for genetic algorithms, in: IEEE International Conference on Evolutionary Computation, 1996, pp. 798–803.
[31] M. Pluhacek, R. Senkerik, A. Viktorin, T. Kadavy, Particle swarm optimization with single particle repulsivity for multi-modal optimization, in: International Conference on Artificial Intelligence and Soft Computing, 2018, pp. 486–494.
[32] B.Y. Qu, J.J. Liang, P.N. Suganthan, Niching particle swarm optimization with local search for multi-modal optimization, 197, 2012, pp. 131–143.
[33] B.Y. Qu, P.N. Suganthan, Multi-objective evolutionary algorithms based on the summation of normalized objectives and diversified selection, Inf. Sci. (Ny) 180 (17) (2010) 3170–3181.
[34] B.Y. Qu, P.N. Suganthan, S. Das, A distance-based locally informed particle swarm model for multimodal optimization, IEEE Trans. Evol. Comput. 17 (3) (2013) 387–402.
[35] H.E. Ran, Y.J. Wang, Q. Wang, J.H. Zhou, H.U. Chen-Yong, An improved particle swarm optimization based on self-adaptive escape velocity, J. Soft. 16 (12) (2005).
[36] I.L. Schoeman, A.P. Engelbrecht, A Parallel Vector-Based Particle Swarm Optimizer, Springer Vienna, 2005.
[37] I.L. Schoeman, A.P. Engelbrecht, Using vector operations to identify niches for particle swarm optimization, in: 2004 IEEE Conference on Cybernetics and Intelligent Systems, vol.1, 2005, pp. 361–366.
[38] G. Schwarz, Estimating the dimension of a model, Ann. Stat. 6 (2) (1978) 15–18.
[39] Y. Shang, Group consensus of multi-agent systems in directed networks with noises and time delays, Int. J. Syst. Sci. 46 (14) (2015) 2481–2492.
[40] Y. Shang, A combinatorial necessary and sufficient condition for cluster consensus, Neurocomputing 216 (2016) 611–616.
[41] Y. Shang, B. Roland, Influence of the number of topologically interacting neighbors on swarm dynamics, Sci. Rep. 4 (8) (2014) 4184.
[42] P.S. Shelokar, P. Siarry, V.K. Jayaraman, B.D. Kulkarni, Particle swarm and ant colony algorithms hybridized for improved continuous optimization, Appl. Math. Comput. 188 (1) (2007) 129–142.
[43] K. Veeramachaneni, T. Peram, C. Mohan, L.A. Osadciw, Optimization using particle swarms with near neighbor interactions, in: International Conference on Genetic and Evolutionary Computation: Parti, 2003, pp. 110–121.
[44] X. Yin, N. Germay, A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization (1993) 450–457.
[45] V.F. Yu, P. Jewpanya, C.J. Ting, A.A.N.P. Redi, Two-level particle swarm optimization for the multi-modal team orienteering problem with time windows, Appl. Soft Comput. 61 (2017).
[46] W.J. Yu, J.Y. Ji, Y.J. Gong, Q. Yang, J. Zhang, A tri-objective differential evolution approach for multimodal optimization, Inf. Sci. (Ny) 423 (2017).
[47] Z.H. Zhan, X.L. Feng, Y.J. Gong, J. Zhang, Solving the flight frequency programming problem with particle swarm optimization, in: Eleventh Conference on Congress on Evolutionary Computation, 2009, pp. 1383–1390.
[48] Z.H. Zhan, J. Zhang, Y. Li, S.H. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst. Man Cybern. Part B Cybern. 39 (6) (2009) 1362.
[49] Z.H. Zhan, J. Zhang, Y.H. Shi, Experimental study on pso diversity, in: Third International Workshop on Advanced Computational Intelligence, 2010, pp. 310–317.
[50] Y. Zhang, Y.J. Gong, H. Zhang, T.L. Gu, J. Zhang, Towards fast niching evolutionary algorithms: a locality sensitive hashing-based approach, IEEE Trans. Evol. Comput. PP (99) (2016). 1–1.