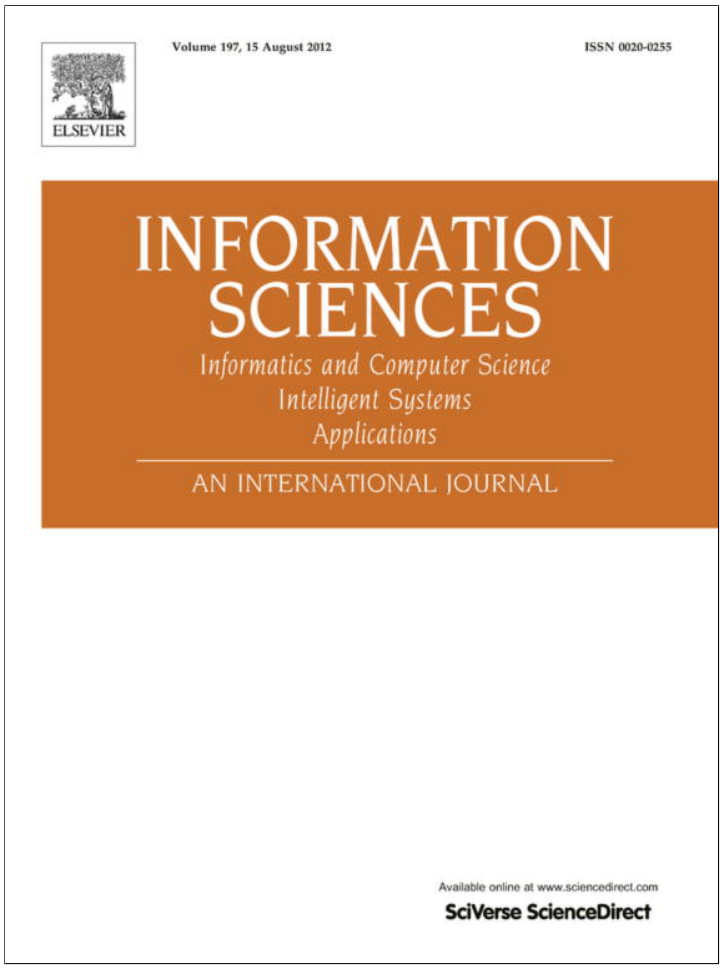

Information Sciences 197 (2012) 38–52

Contents lists available at SciVerse ScienceDirect

Information Sciences

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

A memetic particle swarm optimization algorithm for multimodal optimization problems

Hongfeng Wang [a,b,c,*], Ilkyeong Moon [b,*], Shenxiang Yang [c,d], Dingwei Wang [a,c]

[a] School of Information Science and Engineering, Northeastern University, Shenyang 110819, China
[b] Department of Industrial Engineering, Pusan National University, Pusan, Republic of Korea
[c] State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, China
[d] Department of Information Systems and Computing, Brunel University, Uxbridge, Middlesex UB8 3PH, United Kingdom

ARTICLE INFO

Article history:
Received 9 September 2009
Received in revised form 6 March 2011
Accepted 15 February 2012
Available online 24 February 2012

Keywords:
Multimodal optimization problem
Memetic algorithm
Particle swarm optimization
Local search
Species

ABSTRACT

Recently, multimodal optimization problems (MMOPs) have gained a lot of attention from the evolutionary algorithm (EA) community since many real-world applications are MMOPs and may require EAs to present multiple optimal solutions. In this paper, a memetic algorithm that hybridizes particle swarm optimization (PSO) with a local search (LS) technique, called memetic PSO (MPSO), is proposed for locating multiple global and local optimal solutions in the fitness landscape of MMOPs. Within the framework of the proposed MPSO algorithm, a local PSO model, where the particles adaptively form different species based on their indices in the population to search for different sub-regions in the fitness landscape in parallel, is used for globally rough exploration, and an adaptive LS method, which employs two different LS operators in a cooperative way, is proposed for locally refining exploitation. In addition, a triggered re-initialization scheme, where a species is re-initialized once converged, is introduced into the MPSO algorithm in order to enhance its performance of solving MMOPs. Based on a set of benchmark functions, experiments are carried out to investigate the performance of the MPSO algorithm in comparison with some EAs taken from the literature. The experimental results show the efficiency of the MPSO algorithm for solving MMOPs.

1. Introduction

Hard optimization problems, such as constrained optimization problems [5,23], multi-objective optimization problems [7,34,40], and dynamic optimization problems [3,12,38], have always gained a lot of attention due to their universality in scientific and engineering applications. It is noticeable that many real-world optimization problems are multimodal optimization problems (MMOPs) and may require a solving algorithm to provide multiple optimal solutions in the search space. For example, multiple objects always need to be mapped simultaneously in the machine vision application [4]. For this kind of MMOPs, a solving algorithm is required to obtain all global optima and even locate all, or as many as possible, local optima.

In recent years, investigating the performance of evolutionary algorithms (EAs) for MMOPs has attracted a growing interest from the EA community. However, MMOPs pose serious challenges to traditional EAs since the population tends to converge to a single solution. In order to address this problem, a number of approaches have been developed into EAs for solving MMOPs recently. For example, the *niche* or *speciation* techniques have been developed to first distribute the individuals on multiple different peaks in the solution space and then allow EAs to exploit those peaks simultaneously.

* Corresponding authors. Address: School of Information Science and Engineering, Northeastern University, Shenyang 110819, China (H. Wang).
E-mail addresses: hfwang@mail.neu.edu.cn (H. Wang), ikmoon@pusan.ac.kr (I. Moon).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2012.02.016

Over the years, a class of hybrid EAs, called *memetic algorithms* (*MAs*), have been widely applied for many complex optimization problems, such as scheduling problems [10,20], combinatorial optimization problems [8,35,31], and multi-objective optimization problems [19]. MAs combine EAs with local search (LS) methods and hence are also referred to as genetic local search. In the framework of MAs, LS operators are used to execute further exploitation for the individuals generated by common EA operations (e.g., crossover, mutation, etc.), which is helpful to enhance EA's capacity of solving complicate problems. However, most problems for which MAs have been applied are unimodal problems. MAs have been rarely considered in multimodal environments. Obviously, it is easy to hybridize LS methods with EAs for MMOPs. Therefore, it becomes an interesting research topic to evaluate the performance of MAs for MMOPs.

Similar to EAs, particle swarm optimization (PSO) is also an iterative, population-based optimization technique, which draws its inspiration from the simulation of social behavior of a group of fishes or birds. Due to its features of easy-to-implement and robust adaptability, PSO has been widely applied for many optimization problems [6,9,36]. In the general PSO model, each individual (called *particle*) in the population accomplishes its update based on the best solution (termed as *pbest*) it has found so far, and the best solution (termed as *gbest*) its neighbors have found so far. The principle behind PSO is that each particle owns the learning ability from itself (*pbest*) and its best neighbor (*gbest*). How to choose *gbest* to guide the moving of each particle in the search space is a critical issue. Based on the learning approaches of particles, PSO comes with two versions, i.e., the global version and the local version. In the global PSO model, all particles learn from the best particle in the whole population while in the local version each particle learns from the best particle in its neighborhood. It is obvious that a proper neighborhood topology structure can ensure different particles in the population converge into different optima in the solution space. This property leads to the application of PSO for MMOPs in recent years [14,22].

In this paper, we propose a new memetic PSO (MPSO) algorithm that combines PSO and a LS method for optimization problems with many optima. In the proposed MPSO algorithm, a special PSO neighborhood structure, where the particles are located on a ring-shaped topology and adaptively form different species based on their indices in the population, is proposed for locating multiple optima, and an adaptive LS operator, which employs two different LS methods in an adaptive cooperation fashion, is used for enhancing the exploitation capacity of the proposed algorithm. In addition, a triggered re-initialization scheme is also integrated into the MPSO algorithm in order to further improve its performance for solving those problems with many optima. Extensive experiments are carried out on a set of benchmark multimodal functions in order to examine the major features of the proposed algorithm and compare its performance with some peer algorithms.

The rest of this paper is organized as follows. In Section 2, the basic principle of PSO is introduced and then relevant work on PSO for MMOPs is reviewed briefly. In Section 3, the proposed MPSO algorithm is described in detail. Section 4 presents the experimental results and analyses. Finally, Section 5 concludes this paper with discussions on the future work.

2. Related work

In this section, we describe the basic principle of PSO and then give a brief review on related work on PSO for MMOPs.

2.1. Principles of PSO

PSO is a stochastic optimization method where a population of individuals (particles) move through the search space [17]. The rules, which govern the movement of particles, are inspired by the social interaction among a school of fishes or a flock of birds in nature. In a PSO model, a particle can be represented by its position and its velocity. At every iteration, each particle in the population can accomplish its updating based on its current velocity and position, the best position found so far by itself, and the best position found so far by any of its neighbors, which can be described as follows:

$$\vec{v}_i^j(t+1) = \omega \vec{v}_i^j(t) + c_1 \xi \left(\vec{p}_i^j(t) - \vec{x}_i^j(t) \right) + c_2 \eta \left(\vec{p}_{gi}^j(t) - \vec{x}_i^j(t) \right) \tag{1}$$

$$\vec{x}_i^j(t+1) = \vec{x}_i^j(t) + \vec{v}_i^j(t+1), \tag{2}$$

where $\vec{v}_i^j(t)$ and $\vec{x}_i^j(t)$ represent the velocity and position of particle i in the jth dimension at iteration t, respectively, $\vec{p}_i^j(t)$ is the position in the jth dimension of the best solution (*pbest*) found so far by particle i, $\vec{p}_{gi}^j(t)$ is the position in the jth dimension of the best solution (*gbest*) found so far by the neighbors of particle i, ω is the inertia weight that controls the degree a particle's previous velocity will be kept, c_1 and c_2 denote the cognitive and social learning factors, respectively, and ξ and η are random variables uniformly distributed in the range [0, 1].

Based on the approach of choosing *gbest*, PSO can be classified into two versions, global and local. In the global version of PSO algorithms, all particles in the population "share" the same *gbest* (the best fitness solution found so far by them). As a result, the population can converge quickly into one optimum in the search space. On the contrast, the local version of PSO only allows each particle to choose its *gbest* from its neighbors, which only comprise part of the whole population, in a given distance space. Therefore, the particles in the population may converge into multiple different optima eventually in the local PSO model. It is obvious that the local PSO model can adapt to the multimodal optimization domain more easily than the global version can do.

In the local PSO model, it is very important to design the neighborhood topology structure, i.e., how to determine the neighbors of a particle. One straightforward approach is to define the local neighborhood according to the real distance between the particles in the solution space, which can be translated into the Hamming distance for the genotype with a binary representation or the Euclidean distance for the genotype with a real-coded representation. If the distance between two particles is lower than a given threshold, they are regarded as neighbors to each other. Obviously, this approach is time-consuming because the distance between a particle and other particles always requires to be re-calculated when it chooses its neighbors. Another approach is to identify the neighboring particles using their indices in the population. The particles are often located on a ring-shaped topology, i.e., the particle with index 1 ($\vec{x}_1$) is the immediate neighbor of the particle with index N ($\vec{x}_N$) where N is the population size. If r_s is the neighborhood's radius, particle i is neighbored by the particles with indices from $((i - r_s)\%N)$ to $((i + r_s)\%N)$, where "%" is the modulo operation.

However, there is a well-known problem in the simple local PSO model: it cannot perform an efficient exploitation to locate the optima with a higher accuracy, although it is capable of detecting the regions of different optima in the search space. Hence, it becomes an interesting research issue to investigate the hybridization of the local PSO model and LS techniques for solving MMOPs, which is the major research topic of this paper.

2.2. PSO for MMOPs

Over the past decades, optimization of multimodal functions has been studied widely by EA researchers [13,30,32] and a lot of strategies, such as the niches technique and the species mechanism, have been introduced into EAs for these problems. However, it is noticeable that most work on MMOPs has focused on genetic algorithms (GAs) [18,21,33,37]. Only in very recent years, PSO has been applied in the multimodal optimization domain [27].

The initial work on PSO for MMOPs was reported by Kennedy [16], where a k-means clustering algorithm is used to identify the centers of different clusters of particles in the population, and then these cluster centers are used to substitute the personal best (*pbest*) or the neighborhood best (*gbest*) for each particle. Brits et al. [1] introduced a *nbest* PSO, where the neighborhood of a particle is defined as its n closest particles in the population according to the Euclidean distance, while the *gbest* of each particle is defined as the average of the positions of its n closest neighbors. The disadvantage in the above two early works is that the best neighbor *gbest* of a particle is not always the best fitness particle in its neighborhood, which may mislead its evolution progress.

The niche technique is a common strategy for EAs for MMOPs. Brits et al. [2] introduced the niche technique into the PSO algorithm, where a subswarm can be created with a particle and its closest neighbor from the main swarm if that particle's fitness shows very little change over a number of iterations, and the subswarms thus generated are used to locate multiple optima in parallel in the search space. The subswarms can merge together, or absorb particles from the main swarm and accomplish their training according to the guaranteed convergence PSO model, while the main swarm can be trained based on the cognitive only PSO model. Zhang et al. [39] proposed an adaptive sequential niche PSO (ASNPSO) algorithm for MMOPs. In ASNPSO, multiple subswarms are created to detect multiple optimal solutions sequentially and the particles in the subswarm that runs later need to modify their fitness based on a penalty function once they are determined to locate one of the achieved optima in order to avoid all subswarms from converging to one or several certain optimal solutions. A hill valley function is designed to determine whether or not two points of the search space belong to a peak of the multimodal function through calculating the fitness of several interior sample points between these two points. The disadvantage of this algorithm is that the hill valley detection may consume too many additional evaluations. The authors used a sample array with the size of five in their experiments, which means that one hill valley detection may require five fitness evaluations additionally.

Recently, a species mechanism, which was proposed by Li et al. [18] for improving the performance of genetic algorithms (GAs) for MMOPs, has also been extended into PSO. Parrott and Li [25] proposed a species-based PSO (SPSO) algorithm, where a species can be constructed with its species seed, i.e., the fittest particle in the species, and all particles that fall within a preset Euclidean distance from the species seed. At each iteration step of running SPSO, different species seeds can be identified to form multiple species to search for multiple optima in parallel and then used as the *gbest* for the particles in different species accordingly. In addition, a scheme of removing redundant duplicate particles in a species is also designed for further improving the performance of SPSO.

3. The proposed MPSO algorithm

In this section, we describe three major operations of the proposed MPSO algorithm in detail, including the new local PSO model, the adaptive LS method, and the triggered re-initialization scheme.

3.1. PSO topology structure

The local PSO model is considered for MMOPs in this paper since the local version can adapt better to these problems than the global one (see Section 2.1). However, how to choose the *gbest* for each particle in the population is a major question to answer in the local PSO model.

Parrott and Li [25] defined the local neighborhood in the Euclidean distance space, where the particle with the best fitness is firstly selected as the species seed, and then all particles that fall within a given radius measured in the Euclidean distance from the species seed are classified into the same species, and finally all members in the same species choose the species seed as their *gbest*. The course is iterated until all particles in the population are identified (either as a species seed or a species member). It is easy to see that many evaluations of the Euclidean distance are required during this identification course. Petalas et al. [26] arranged all particles in the population on a ring topology and decided the neighbors of each particle according to a neighborhood radius measured by the population index distance between two particles. The *gbest* of each particle can be determined by the fittest particle among its neighbors. Although this scheme seems very simple, the interaction among the particles may eventually lead to the population converging into a few optima or even only one optimum, which is not expected when solving MMOPs.

Inspired by the aforementioned works, a robust algorithm of choosing *gbest* for each particle in the population is proposed in this paper, where the particles adaptively form multiple different species based on their population indices, and then the species seed (the best fitness particle in the species) is chosen as the *gbest* of each member in the species accordingly.

The choice algorithm is performed at each iteration step, as shown in Fig. 1, where S_0 denotes the set of solutions achieved by the MPSO algorithm so far (see the detailed discussion in the next section), S denotes the set of species seeds, r_s is a parameter of controlling the size of a species, and r_0 is a parameter of checking whether a particle locates in an achieved optimum. At the beginning, all achieved solutions in S_0 are first copied into the set S and the status of each particle in the population P is initially set as *unprocessed*. The best unprocessed particle $\vec{x}_k$ is first selected from P to check whether it will be chosen as a species seed or not. If $\vec{x}_k$ falls within the radius r_0 from any solution $\vec{s}$ in S, it cannot become a species seed and will be re-initialized immediately with a random position and velocity; otherwise, $\vec{x}_k$ will be chosen as a new species seed and added into S, and a new species will be constructed by all unprocessed particles whose indices are within the range $[k - r_s, k + r_s]$.

It is easy to understand that the above choice algorithm aims at forming multiple species, which can explore different regions of the search space in parallel, according to the indices of particles in the population. Fig. 2 provides an example to illustrate how this algorithm works. In this case, a population consists of six particles, among which s_1, s_2, and s_3 are the first, second, and third fittest particles, respectively. Two species, dominated by s_1 and s_3, respectively, will be constructed after the choice algorithm is applied, while s_2 cannot be chosen as a species seed since the distance between s_2 and s_1 is shorter than r_0, which means that s_2 may be located in the same peak as s_1. Note also that particle p only belongs to the species dominated by s_1, which means that a particle can only be dominated by the fitter species seed in order to avoid the interaction between two different species if they have their radii overlapped.

The aforementioned species construction method is expected to decrease the number of Euclidean distance calculations. In the neighborhood scheme of the local PSO model that computes all distances among the particles at each iteration, the number of Euclidean distance calculations is $(s\_size - 1)!$ where *s_size* denotes the size of the whole swarm in this paper,

```
Procedure Algorithm of choosing gbest:
begin
  S := S_0;
  set seed := true;
  set the status of each particle in P as unprocessed;
  while (there are unprocessed particles in P) do
    get the best unprocessed particle x_k ∈ P;
    for each species seed s ∈ S do
      if d(x_k, s) < r_0 then
        initialize x_k randomly;
        set the status of x_k as processed;
        set seed := false;
        break;
      end if
    end for
    if (seed = true) then
      S := S ∪ {x_k};
      for each unprocessed particle x with an index in [k − r_s, k + r_s] do
        set x_k as x's gbest;
        set x's status as processed;
      end for
    end if
  end while
end
Denotations:
  S: the set of selected species seeds;
  S_0: the set of solutions achieved by the algorithm so far;
  P: the current population;
  d(x, y): the Euclidean distance between two particles x and y;
  r_s: a parameter of controlling the size of a species;
  r_0: a parameter of checking whether a particle locates in an achieved optimum.
```

Fig. 1. Pseudo-code for the algorithm of choosing *gbest* for each particle.

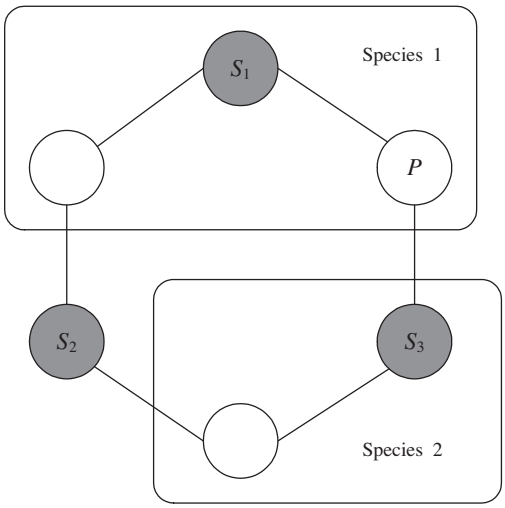

Fig. 2. An example of how to form the species in a population of six particles ($r_s = 1$).

and in the species method in SPSO [25], the maximal number of Euclidean distance calculations is $(s\_size - 1)!$, while the corresponding number is $(s\_size - 2 * r_s - 1)!$ in our proposed species method considering that there is always a species with the size of $2 * r_s + 1$ dominated by the fittest particle in the population to be constructed when our proposed species method is applied in each iteration. In the example of Fig. 2, the number of Euclidean distance calculations is 3 using our proposed method, while it is at least 5 using the species method in SPSO [25].

3.2. Local search

In the above local PSO model, multiple species are used to explore different regions in the search space during the iterative process of the algorithm. Obviously, the species should be constructed as many as possible in order to ensure that each optimal solution in MMOPs is achieved by the algorithm. This means that the parameter r_s cannot be set to a very large value. However, it is also noticeable that the smaller the size of a species, the less efficient its exploitation of achieving the optima with a high accuracy. To enhance the exploitation capacity of the aforementioned local PSO model, LS methods are used to improve the quality of species seeds at each iteration.

The hybridization of EAs and LS (i.e., MA) has been widely investigated by the EA community in recent years. In MAs, EA operations are used for global rough exploration and LS methods are used for local refining exploitation. The LS procedure in the context of PSO can be regarded as the particle making one iterative local move, while the moving from the current solution to a candidate solution will be accepted if the candidate solution has a better fitness.

Here, two different LS operators are considered in our proposed algorithm, which are described as follows.

(1) Cognition-Based Local Search (*CBLS*): This LS operator is designed according to a special PSO model. Kennedy [15] introduced two special PSO models, which are defined by omitting or restricting components of the velocity formula. Dropping the social component results in the *cognition-only model*, whereas dropping the cognitive component defines the *social-only model*. It is clear that the cognition-only model can help enhance the exploitation capacity of a particle to its own best *pbest*, which is the main motivation of designing the *CBLS* operator. The *CBLS* procedure, assuming a maximization problem, can be expressed by the pseudo-code shown in Fig. 3, where *ls_num* denotes the step size of an LS operation and $\vec{p}$ denotes the *pbest* of a selected particle $\vec{x}$.

(2) Random Walk with Direction Exploitation (*RWDE*): This LS operator is an iterative, stochastic optimization method that generates a sequence of approximations of the optimizer [26]. Rather than directing the moving of a particle by its *pbest* in the *CBLS* operator, the particle always assumes a random vector as its search direction at each LS step when *RWDE* is applied. The *RWDE* procedure is outlined in Fig. 4, where λ denotes a prescribed scalar step length.

From the pseudo-codes in Figs. 3 and 4, it can be seen that the two LS operators adopt different moving ways respectively. The *CBLS* operator can direct a particle's search towards its *pbest*, which is based on the idea that *pbest* may be closer to the optimum. The *RWDE* operator aims to execute an efficient random search around the current solution through decreasing the step length gradually.

It has been reported in the literature [24,28] that the effect of LS is problem-dependent, that is, a single LS operator makes MAs efficient for some classes of problems, or even only improves the performance of MAs in several iterative steps during the running. Therefore, an adaptive LS method, where *CBLS* and *RWDE* are both employed to work together to accomplish a shared optimization goal in an efficient cooperation fashion, is proposed in the proposed MPSO algorithm, which is illustrated in Fig. 5.

Procedure $CBLS(\vec{x})$:
begin
  initialize a velocity vector $\vec{v}'$ within a small range;
  for $j := 1$ **to** $ls\ num$ **do**
    $\vec{x}' := \vec{x}$;
    $\vec{x}' := \vec{x}' + \omega\vec{v}' + c_1\xi(\vec{p} - \vec{x}')$;
    if $f(\vec{x}') > f(\vec{x})$ **then** $\vec{x} := \vec{x}'$ **end if**
  end for
end;
Denotations:
  $\vec{x}$: the selected particle for local improvement
  $ls\ num$: the step size of an LS operation
  $\vec{p}$: the *pbest* of a selected particle $\vec{x}$

Fig. 3. Pseudo-code for the *CBLS* method.

Procedure $RWDE(\vec{x})$:
begin
  for $j := 1$ **to** $ls\ num$ **do**
    initialize a unit-length random vector $\vec{v}$;
    $\vec{x}' := \vec{x}' + \lambda\vec{v}$;
    if $f(\vec{x}') > f(\vec{x})$ **then** $\vec{x} := \vec{x}'$;
    else $\lambda := \lambda/2$;
    end if
  end for
end
Denotations:
  λ: a prescribed scalar step length
  Other parameters are the same as those for $CBLS$

Fig. 4. Pseudo-code for the *RWDE* method.

Procedure Adaptive LS method:
begin
  if $rand() < p_{ls}$ **then**
    if $d(\vec{x}, \vec{p}) < r_1$ **then** $RWDE(\vec{x})$;
    else $CBLS(\vec{x})$;
    end if
  end if
end
Denotations:
  $rand()$: a random generated number in (0,1)
  p_{ls}: the probability of executing LS method
  r_1: a very small positive number
  Other parameters are the same as those for $CBLS$

Fig. 5. Pseudo-code for the adaptive LS method.

In this adaptive LS method, the particle that is selected for local improvement first requires to check the Euclidean distance between its *pbest* and itself. If the distance is less than a preset constant, the *RWDE* operation will be applied on it; otherwise, the *CBLS* operation will be executed on it. It is easy to understand that *CBLS* actually becomes a random search when the particle approaches too closely to its *pbest* according to the pseudo-code in Fig. 3. In addition, the LS operation is also executed by a probability p_{ls} in this adaptive LS strategy. Ideally, p_{ls} should be set to a very large value if the LS operation can improve the performance of individuals significantly, while it should become very small if LS is not very helpful.

In this paper, we propose a robust scheme to decide the value of p_{ls}. Let $n_v(t-1)$ and $n_T(t-1)$ denote the actual number of valid local moving steps and total local moving steps carried out during iteration $t-1$, respectively. Then, the value of $p_{ls}(t)$ is calculated as follows:

$$sign(t) = \begin{cases} 1, & \frac{n_v(t-1)}{n_T(t-1)} < \delta \\ 0, & \frac{n_v(t-1)}{n_T(t-1)} = \delta \\ -1, & \frac{n_v(t-1)}{n_T(t-1)} > \delta \end{cases} \tag{3}$$

```
Procedure Proposed memetic PSO algorithm:
begin
  generate and evaluate an initial population P randomly;
  repeat
    determine the species seeds for the current population P (see Figure 1);
    execute LS operation (see Figure 5) for each species seed;
    update each particle in P according to Eqs. (1) and (2) respectively;
    re-initialize each converged species (see Section 3.3);
  until a termination condition is met
end
```

Fig. 6. Pseudo-code for the proposed MPSO algorithm.

$$p_{ls}(t) = min\left\{p_{ls}^{max}, max\left\{\beta^{sign(t)} * p_{ls}(t-1), p_{ls}^{min}\right\}\right\} \tag{4}$$

where $\delta \in (0, 1)$ is a preset constant that acts as the threshold of changing p_{ls}: decreasing, increasing, or neither, $\beta \in (0, 1)$ controls the decreasing or increasing speed of p_{ls}, and p_{ls}^{max} and p_{ls}^{min} are predefined maximal and minimal value of p_{ls}, respectively. From these formula, it can be seen that this adaptive scheme is designed to avoid executing too many useless LS operations.

3.3. Triggered re-initialization

Based on the design of the PSO neighborhood topology structure in Section 3.1, the population can be divided into a number of species to locate multiple optima in parallel. Obviously, the number of species, whose value is dependent on the swarm size and the species size, is expected to be larger than the number of optima in the search space in order to ensure the algorithm to find all optimal solutions. However, this expectation cannot be achieved since the number of optima is always unknown in advance, while the algorithm parameters need to be predefined. Once there are numerous optima in the fitness landscape, it is obvious that this local PSO model cannot ensure obtaining all optimal solutions.

To address this problem, with the inspiration of re-initialization for converged population in EAs [11], a triggered re-initialization scheme is proposed for MPSO and discussed in this section. In this scheme, each species will be checked whether it has converged into one optimum at each iteration. If a species converges into one optimum, the re-initialization of that species will be triggered and its species seed will be copied into a solution set, i.e., S_0 in Fig. 1, as an achieved solution. This strategy seems similar to the sequential niche technique in [39] except that it employs the re-initialization mechanism, rather than the penalty function, in order to keep the algorithm from exploiting the achieved optima.

The next question to be answered is how to design the triggered condition. Here, a fitness based diversity index ξ is used to judge whether a species converges into one optimal solution or not. Let f_{seed} and f_{ave} denote the best and average fitness among the fitness values of a species, respectively. The index ξ can be calculated as follows:

$$\xi = min\left\{\left|\frac{f_{ave} - f_{best}}{f_{best}}\right|, 1\right\} \tag{5}$$

Obviously, ξ can be regarded as a measurement of the convergence degree of a species. If $\xi \to 1$, the species is far from converged, while the species is approaching convergence if $\xi \to 0$. The triggered re-initialization of a species can be designed using the index ξ. If the value of ξ corresponding to a species is lower than a certain threshold θ, this species will be re-initialized and the species seed is copied into S_0 as an achieved solution.

Another problem that follows when this triggered re-initialization scheme is integrated into our algorithm is that the triggered generator may mistake a non-optimal solution as an optimum in some cases. In the proposed PSO model, species can be constructed adaptively, which means that different species can contain different number of individuals at each iteration. When there is only one individual in one species, the value of ξ is always equal to 0. It is obvious that such species cannot achieve one optimal solution. Therefore, only a full species, where there are the maximum allowable number $(2 * r_s + 1)$ of individuals, will be checked to see whether it will be re-initialized or not at each iteration.

Based on the above discussion, an adaptive LS method, where *CBLS* and *RWDE* are employed in a cooperative fashion, and a triggered re-initialization scheme, which enables the algorithm to achieve the optima sequentially, are integrated into the local PSO model with the neighborhood topology structure described in Section 3.1 to solve MMOPs in this paper. The proposed MPSO algorithm is summarized in Fig. 6.

4. Experimental study

In this section, we first introduce ten test functions that are well studied in the literature and then examine the performance of MPSO in comparison with some peer algorithms on these test functions.

4.1. Test functions

The ten test functions that are widely used in the literature to examine the performance of algorithms on locating multiple global and local optima are described as follows:

Test Problem 1. The problem is defined as

$$F1(x) = \sin^6(5\pi x), \tag{6}$$

where $0 \leqslant x \leqslant 1$ and there are five global maxima at $x = 0.1$, $x = 0.3$, $x = 0.5$, $x = 0.7$, and $x = 0.9$, respectively.

Test Problem 2. The problem is defined as

$$F2(x) = \exp\left(-2\log(2) \times \left(\frac{x-0.1}{0.8}\right)^2\right) \times \sin^6(5\pi x), \tag{7}$$

where $0 \leqslant x \leqslant 1$ and there are one global maximum at $x = 0.1$ and four local maxima at $x = 0.3$, $x = 0.5$, $x = 0.7$, and $x = 0.9$, respectively.

Test Problem 3. The problem is defined as

$$F3(x) = \sin^6(5\pi(x^{3/4} - 0.05)), \tag{8}$$

where $0 \leqslant x \leqslant 1$ and there are five global maximum at $x \approx 0.08$, $x \approx 0.25$, $x \approx 0.45$, $x \approx 0.68$, and $x \approx 0.93$, respectively.

Test Problem 4. The problem is defined as

$$F4(x) = \exp\left(-2\log(2) \times \left(\frac{x-0.08}{0.854}\right)^2\right) \times \sin^6(5\pi(x^{3/4} - 0.05)), \tag{9}$$

where $0 \leqslant x \leqslant 1$ and there are one global maximum at $x \approx 0.08$ and four local maxima at $x \approx 0.25$, $x \approx 0.45$, $x \approx 0.68$, and $x \approx 0.93$, respectively.

Test Problem 5. The problem is defined as

$$F5(x, y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2, \tag{10}$$

where $-6 < x,y < 6$ and there are four global maxima at $(x, y) = (3.0, 2.0)$, $(x, y) \approx (-3.78, -3.28)$, $(x, y) \approx (3.58, -1.85)$, and $(x, y) \approx (-2.81, 3.13)$, respectively.

Test Problems 6–8 (Shekel Functions). The problem is defined as

$$S_{4,n} = -\sum_{i=1}^{n}[\cos(x - a_i)^T(x - a_i) + c_i]^{-1}, \tag{11}$$

where $x = \{x_1, x_2, x_3, x_4\}$ with $x_j \in (0, 10)$, $j = 1, \ldots, 4$, $a = \{\{4.0, 4.0, 4.0, 4.0\}, \{1.0, 1.0, 1.0, 1.0\}, \{8.0, 8.0, 8.0, 8.0\}, \{6.0, 6.0, 6.0, 6.0\}, \{3.0, 7.0, 3.0, 7.0\}, \{2.0, 9.0, 2.0, 9.0\}, \{5.0, 5.0, 3.0, 3.0\}, \{8.0, 1.0, 8.0, 1.0\}, \{6.0, 2.0, 6.0, 2.0\}, \{7.0, 3.6, 7.0, 3.6\}\}$, and $c = \{0.1, 0.2, 0.2, 0.4, 0.4, 0.6, 0.3, 0.7, 0.5, 0.5\}^T$. Three functions are tested in this paper, denoted $S_{4,5}$, $S_{4,7}$, and $S_{4,10}$, where there are five, seven, and ten minima, respectively.

Test Problem 9 (Shubert Function). The problem is defined as

$$F9(x, y) = \left(\sum_{i=1}^{5} j\cos((j+1)x + j)\right) \cdot \left(\sum_{i=1}^{5} j\cos((j+1)y + j)\right), \tag{12}$$

where $-10 < x, y < 10$ and there are eighteen global minima.

Test Problem 10 (Foxhole Function). The problem is defined as

$$F10(x, y) = 500 - \frac{1}{0.002 + \sum_{i=0}^{24} 1/(1 + i + (x - a(i))^6 + (y - b(i))^6)}, \tag{13}$$

where $-65.536 \leqslant x, y \leqslant 65.536$, $a(i) = 16((i \bmod 5) - 2)$, $b(i) = 16((\lfloor i/5 \rfloor) - 2)$, and there are 25 global maxima.

4.2. Experimental setup

Experiments were carried out to compare the proposed algorithm with several state-of-the-art algorithms on the test functions described above. The following abbreviations represent these algorithms considered in this paper.

- MSPO: our proposed algorithm which hybridizes the adaptive LS method and the triggered re-initialization scheme into the local PSO model described in Section 3.1;
- ASNPSO: the PSO algorithm proposed by Zhang et al. [39], where the sequential niche technique and multi-swarm scheme are hybridized for MMOPs;

- SPSO: the species-based PSO algorithm proposed by Parrott and Li [25], where species can be constructed in the Euclidean distance space.
- RWMPSO: the memetic PSO algorithm proposed by Patalas et al. [26], where the neighborhood topology is used, similar to the PSO structure described in Section 3.1, and the RWDE operations are applied to refine the best particle and some particles selected by probability from the swarm.

The following parameters are used for all algorithms: the cognitive and social learning factors c_1 and c_2 were both set to 1.4962, the inertia weight ω was set to 0.72984, and the velocity of a particle was always confined within the range $[-V_{MAX}, V_{MAX}]$, where $-V_{max}$ and V_{max} are the lower and upper bounds of the velocity of a particle. The population size in MPSO, SPSO, and RWMPSO was set to 30 for the first five test functions F1-F5, to 50 for the following three test functions F6-F8, and to 100 for the last two functions F9 and F10. The size of each subswarm in ASNPSO was set to 10 for all test functions. The special parameters in the MPSO algorithm were set as follows: $r_s = 2$ $ls\_num = 5$, $r_1 = 0.01$, $p_{ls}^{max} = 1.0$, $p_{ls}^{min} = 0.1$, $\beta = 0.5$, and $\theta = 0.000001$. The value of r_0 was set normally to half of the distance between two closest optima (the similar setting was also used in [25]), considering that the exact number of optima and the distances between them are known. Other genetic operators and parameters in the peer algorithms used were the same as their original settings.

Since it is always assumed that the global and local optima of all test functions are known a prior, the performance of an algorithm is evaluated by the following two measurements.

(1) *Accuracy*: In order to examine the accuracy of an algorithm, that is, how close the fittest solutions achieved by the algorithm are to all known optima, each algorithm was run for a fixed number of evaluations (the number of fitness evaluations is considered as the time measurement of an algorithm in this paper). Note that each known optimum corresponds only to a different solution which is the fittest particle among S in the final iteration of the algorithm. More mathematically, accuracy can be calculated by taking the average of the relative fitness differences between all known optima to their closest achieved solutions as follows:

$$\text{accuracy} = \frac{1}{|S^*|}\sum_{i=1}^{|S^*|}\left|\frac{f(s_i^*) - f(s_i)}{f(s_i^*)}\right|, \tag{14}$$

where S^* denotes the set of all known optima and $|S^*|$ returns the number of known optima. The pair of s_i^* and s_i denote that for each optimum $s_i^* \in S^*$, there is correspondingly a closest solution $s_i \in S$ to s_i^*, where S represents the set of solutions achieved by the algorithm. Obviously, s_i can be identified from S via checking whether the distance between s_i^* and any solution of S is lower than a preset radius. If s_i^* is not achieved by the algorithm (which means it is not within r_0 of any solution of S in this paper), $f(s_i)$ is simply set to 0. Since both global and local optima are considered in this study, the relative fitness difference $\left(\left|\frac{f(s_i^*)-f(s_i)}{f(s_i^*)}\right|\right)$, rather than the absolute difference $(|f(s_i^*) - f(s_i)|)$, is used in order to normalize this measurement into the same level for either global or local optima.

(2) *Convergence Speed*: In order to examine the convergence speed of an algorithm, that is, how fast an algorithm can achieve all optima, each algorithm was run until all optima were achieved at the expected level of accuracy. The speed measurement is defined by the number of evaluations required to achieve the expected accuracy. In this study, one optimum $s_i^* \in S^*$ is said to be found when there exists one solution s_i in S that falls within the distance radius r_0 from s_i^* and $\left|\frac{f(s_i^*)-f(s_i)}{f(s_i^*)}\right|$ is lower than a preset accuracy acceptance threshold ϵ. If all optima could not be found until the maximum number of evaluations allowed was reached, the maximum allowable number of evaluations was recorded.

In addition, we also measure the performance of algorithms in terms of the success rate, which is defined as the proportion of the achieved optima in relation to all known optima, in this study.

4.3. Basic experimental results and analysis

In the basic experimental study, we compare the capability of the four peer algorithms in locating multiple optima. In particular, MSPO, ASNPSO, SPSO, and RWMPSO were executed 30 independent runs with the same random seeds and the experimental results are reported via averaging across the total 30 runs. To measure the performance of algorithms, the expected accuracy (i.e., the accuracy threshold) ϵ was set to 0.0001 and the maximum allowable number of evaluations was set to 30,000 for the first five test functions F1–F5, to 50,000 for the following three test functions F6–F8, and to 100,000 for the last two functions F9 and F10 for all algorithms. The experimental results with respect to the accuracy performance and the success rate are presented in Tables 1 and 2, respectively. In Table 1, the *t*-test column denotes the corresponding statistical results of comparing the MPSO algorithm with the corresponding peer algorithm by the one-tailed *t*-test with 58 degrees of freedom at a 0.05 level of significance, considering that the sample size of the experimental results is 30 for both compared algorithms and the statistical results obey the *t* distribution. The *t*-test result is shown as "*s*+" or "+" when MPSO is significantly better than, or insignificantly better than the corresponding algorithm, respectively. From Tables 1 and 2, several results can be observed and are analyzed below.

Firstly, a prominent result is that MPSO outperforms SPSO, ASNPSO, and RWMPSO on all test functions. For example, MPSO can always achieve all global and local optima on all functions, while SPSO can find all global optimal solutions on several test functions (e.g., F1, F3, F5, and F6), ASNPSO cannot achieve all optima on the last five functions, and RWMPSO can only achieve all optima on F6. In addition, MPSO can always locate the achieved optima with the highest accuracy on all test functions, i.e., the accuracy performance of MPSO is significantly better than the peer algorithms, as shown from the *t*-test results in Table 1. In MPSO, the proposed topology structure, where multiple species can be constructed adaptively according to the indices of particles in the population at each iteration, can help the algorithm explore different areas in the search space efficiently and the proposed adaptive LS method, which employs two different LS operators in a cooperative fashion, can execute a robust exploitation to each species seed. In addition, the triggered re-initialization scheme can provide MPSO the capacity of exploring new optima and prevent MPSO from re-exploring the achieved optima. The experimental results of MPSO for the test functions validate our expectation of the proposed algorithm for MMOPs.

The good performance of MPSO over SPSO, ASNPSO, and RWMPSO can be further observed in the experimental results with respect to the convergence speed, as shown in Table 3, where "–" means a corresponding algorithm did not achieve all optima before the maximum number of allowable evaluations was reached on a test problem. MPSO is able to achieve all optima (either global or local) in the expected accuracy using the least number of evaluations for all test problems. For example, the average number of evaluations required by MPSO is 1324, 1828, 1269, 1686, 1679, 14,472, 23,594, and

Table 1
Experimental results with respect to the accuracy of MSPO and peer algorithms on the test functions.

Funct.	MPSO	SPSO		ASNPSO		RWMPSO	
	Accuracy	Accuracy	*t*-Test	Accuracy	*t*-Test	Accuracy	*t*-Test
F1	7.86E−16	1.40E−11	*s*+	2.92E−07	*s*+	8.00E−02	*s*+
F2	1.57E−13	5.50E−05	*s*+	8.08E−09	*s*+	8.00E−01	*s*+
F3	4.76E−15	5.08E−12	*s*+	3.67E−07	*s*+	1.60E−01	*s*+
F4	1.39E−14	5.68E−06	*s*+	3.80E−07	*s*+	8.00E−01	*s*+
F5	3.70E−14	1.21E−12	*s*+	2.13E−10	*s*+	1.50E−01	*s*+
F6	7.23E−10	1.34E−09	+	1.97E−01	*s*+	2.92E−07	*s*+
F7	2.31E−06	1.04E−03	*s*+	3.63E−01	*s*+	2.92E−07	*s*+
F8	5.04E−06	1.53E−02	*s*+	4.90E−01	*s*+	2.92E−07	*s*+
F9	3.19E−07	2.52E−02	*s*+	2.02E−02	*s*+	9.44E−01	*s*+
F10	5.08E−13	1.23E−01	*s*+	2.85E−01	*s*+	9.60E−01	*s*+

Table 2
Experimental results with respect to the success rate of MSPO and peer algorithms on the test functions.

Funct.	MSPO (%)	SPSO (%)	ASNPSO (%)	RWMPSO (%)
F1	100.00	100.00	100.00	92.00
F2	100.00	97.33	100.00	20.00
F3	100.00	100.00	100.00	83.33
F4	100.00	99.33	100.00	20.00
F5	100.00	100.00	100.00	81.67
F6	100.00	100.00	79.33	100.00
F7	100.00	99.05	63.33	5.56
F8	100.00	95.00	51.00	4.00
F9	100.00	97.22	82.22	5.56
F10	100.00	85.47	71.47	4.00

Table 3
Experimental results with respect to the convergence speed of MSPO and peer algorithms on the test functions.

Funct.	MPSO	SPSO		ASNPSO		RWMPSO	
	Conv. speed	Conv. speed	*t*-Test	Conv. speed	*t*-Test	Conv. speed	*t*-Test
F1	1324	2143	*s*+	15,948	*s*+	25,836	*s*+
F2	1828	3524	*s*+	19,465	*s*+	–	*s*+
F3	1269	2330	*s*+	16,680	*s*+	31,745	*s*+
F4	1686	2532	*s*+	14,451	*s*+	–	*s*+
F5	1679	2892	*s*+	29,860	*s*+	21,354	*s*+
F6	14,472	15,746	*s*+	–	*s*+	18,131	*s*+
F7	23,594	28,096	*s*+	–	*s*+	–	*s*+
F8	42,213	42,851	+	–	*s*+	–	*s*+
F9	44,086	–	*s*+	–	*s*+	–	*s*+
F10	30,950	–	*s*+	–	*s*+	–	*s*+

42,213 on the first eight functions, respectively, when the expected accuracy threshold ϵ was set to 0.0001. The t-test results, which also denote the corresponding statistical results of comparing MPSO with the corresponding peer algorithms by the one-tailed t-test with 58 degrees of freedom at a 0.05 level of significance, show that the results of MPSO are significantly better than the corresponding results of the three peer algorithms on the first eight functions, respectively. For the last two functions, only MPSO can achieve all optimal solutions successfully before the maximum number of allowable evaluations was reached, while the other three algorithms failed to do so.

Secondly, SPSO exhibits a good performance in achieving the global optima on most test functions, but performs worse in locating the local optima. SPSO can find all optima successfully on F1, F3, F5, F6, and F7, while the success rate is 97.33% on F2 and 99.33% on F4, respectively, which means that SPSO cannot locate some local optima with the lower fitness accurately. The reason lies in that the species in SPSO are constructed in an imbalanced way, that is, different species may have different number of members. In SPSO, the species seeds are determined in turn according to the decreasing order of fitness, and then each species consists of all the non-dominated particles in the population that fall within a predefined radius from the species seed. It is easy to see that the species dominated by the species seeds with the higher fitness always compose of much more members than those dominated by the lower fitness species seeds, which is the reason that the local optima with the lower fitness cannot be achieved by SPSO accurately. However, the size of the species in MPSO, constructed based on the index of particles in the population, is limited by the species radius r_s and the particles are initialized when they are determined to approach very closely to the achieved optima located by the existing species. This mechanism enables the algorithm to make a balance of exploitation among different species as much as possible. The experimental results indicate the efficiency of the MPSO algorithm for locating both global and local optima.

In addition, when SPSO is applied for the test functions with numerous optima, the species cannot locate all optima due to the limitation of the population size. For example, the success rate of SPSO is 97.22% on F9 and 85.47% on F10, respectively (see Table 2) and SPSO cannot achieve all optima in the expected accuracy when evaluating its convergence speed (see Table 3). These experimental results show the necessity and effect of the triggered re-initialization scheme for MMOPs.

Thirdly, ASNPSO performs a little poorly in the experiments, although it can locate all of the global and local optima with the success rate of 100.00% on the first five test functions. This happens because the hill valley detection in ASNPSO, which is a special operator to check whether a particle locates one of the achieved peaks (or optima), consumes too many evaluations considering that the time is measured in terms of evaluations in our experiments. This is further validated by the experimental results in Table 3, where ASNPSO always requires much more evaluations than MPSO and even SPSO in order to achieve all optima on the first five functions. Actually, ASNPSO may even locate all optima for the last five functions if the maximum allowable number of evaluations was set to a very large value. The similar results were reported in Zhang et al. [39].

Finally, RWMPSO performs the worst in the experiments, where it only achieves several optima or even only one single optimum on most test functions. The reason lies in that its population would converge into a few optima or even one optimum gradually due to the interaction among particles, which indicates that our proposed local topology structure is a good choice when PSO is applied for solving those problems with multiple optima.

4.4. Experimental study on the effect of major operators in MPSO

As discussed before, MPSO hybridizes a local PSO model where the particles can form different species based on their population indices, an adaptive LS method which employs two different LS methods for refining the quality of species seeds in a cooperative fashion with a probability, and a triggered re-initialization scheme that re-initializes all particles in a converged species randomly and stores the species seed in a solution set as an achieved optimum. The following sets of experiments were carried out to examine the effect of these major components upon the performance of MPSO and analyze the sensitivity of some key parameters in the three major components on five test functions F1, F2, F5, F6, and F10.

4.4.1. The effect of the parameter r_s

In the basic experiments in Section 4.3, RWMPSO fails to locate multiple optima, while MPSO can achieve all optima on the test functions. This means that a proper local topology structure is helpful for PSO in multimodal environments. Obviously, the parameter r_s plays a very important role in the proposed PSO model. In this set of experiments, we investigate the effect of r_s on the performance of MPSO. MPSO was run 30 times with r_s set to the values in {1, 2, 5, 10, s_size/2}, respectively, on the five test functions. The other parameters were set to the same values as those in the basic experiments. The experimental results with respect to the accuracy and success rate performance measures are shown in Tables 4 and 5, respectively.

From Table 4, it can be seen that the setting of r_s significantly affects the performance of MPSO. On all test functions, MPSO can find quite a few optima when r_s is set to a too large value, while it can find multiple optima when r_s is set to a small value. This is easy to understand. The smaller the value of r_s, the more the number of species. It is obvious that the number of species can affect the algorithm's capability of exploring multiple sub-regions where the optima are located in the fitness landscape. This is the reason why MPSO can achieve all optima with the success rate of 100% on all test functions when $r_s = 1$ and $r_s = 2$, while it failed when r_s was set to 5, 10, and s_size/2, respectively. However, when r_s was set to a too small value, the size of each species also becomes very small, which can induce the algorithm not to make a sufficient exploitation in the searching regions where the species locate. As shown in Table 5, MPSO can locate the optima with a much

Table 4
Experimental results with respect to the accuracy performance of MSPO with different values of r_s on the test functions.

Funct.	$r_s = 1$	$r_s = 2$	$r_s = 5$	$r_s = 10$	$r_s = s\_size/2$
F1	6.10E−12	7.86E−16	1.47E−01	6.00E−01	7.87E−01
F2	2.90E−11	1.57E−13	9.59E−17	5.33E−02	2.52E−15
F5	1.35E−09	3.70E−14	1.78E−06	5.42E−01	5.33E−01
F6	1.45E−05	1.34E−09	9.48E−10	3.42E−01	5.34E−01
F10	4.44E−11	5.08E−13	2.78E−01	6.40E−01	8.73E−01

Table 5
Experimental results with respect to the success rate of MSPO with different values of r_s on the test functions.

Funct.	$r_s = 1$ (%)	$r_s = 2$ (%)	$r_s = 5$ (%)	$r_s = 10$ (%)	$r_s = s\_size/2$ (%)
F1	100.00	100.00	84.00	40.00	21.00
F2	100.00	100.00	100.00	94.67	100.00
F5	100.00	100.00	99.17	45.83	46.67
F6	100.00	100.00	100.00	62.00	44.00
F10	100.00	100.00	68.53	34.27	12.67

Table 6
Experimental results with respect to the accuracy of MPSO with different LS operators on the test functions.

Funct.	LPSO	MPSO_0	MPSO_1	MPSO_2
F1	3.13E−14	7.86E−16	2.41E−15	6.84E−15
F2	1.10E−12	1.57E−13	2.54E−13	2.78E−13
F5	4.07E−12	3.70E−14	1.10E−12	6.57E−14
F6	1.91E−07	1.34E−09	4.57E−09	2.68E−08
F10	1.89E−12	5.08E−13	8.89E−13	1.41E−13

higher accuracy when $r_s = 2$ than when $r_s = 1$. The optimal setting of r_s for all test functions is $r_s = 2$, which enables MPSO to achieve the best performance with respect to both the accuracy and success rate performance measures.

4.4.2. The effect of different LS operators

Based on the results of the basic experiments, MPSO can always locate the optima with a higher accuracy than the other peer algorithms due to the effect of the proposed LS method. In the following set of experiments, we further investigate the performance of MPSO with different LS operators on the five test functions. In order to make a convenient description of the experimental results, MPSO_0, MPSO_1, and MPSO_2 are used to denote MPSO with both CBLS and RWDE operators, MPSO with only the CBLS operator, and MPSO with only the RWDE operator, respectively, while LPSO denote MPSO without any LS operator. In all algorithms investigated in this set of experiments, the LS operator was always executed by a probability, which can be adjusted according to the same way in the basic experiment. The relevant parameters were set to the same values as those used in the basic experiments. The experimental results with respect to the accuracy performance are presented in Table 6.

From Table 6, the following results can be observed. Firstly, the proper LS technique does help the algorithm achieve the optima more accurately, which can be shown from the results that all MPSO algorithms always perform better than LPSO on the test functions. Secondly, the effect of LS operators is problem-dependent. For example, MPSO_1 outperforms MPSO_2 on F1, F2, and F6, but is beaten by MPSO_2 on F5 and F10. Thirdly, the cooperative mechanism can help MPSO execute more efficient local refinement. In the experiments, MPSO_0 always performs the best on all test functions.

Another feature in our proposed LS method is that the LS operation is executed by a probability. Obviously, the purpose of this scheme is to reduce the number of useless LS operations given that the LS operation may consume many computation time in terms of evaluations. The following set of experiments was carried out in order to examine the effect of the probabilistic execution of LS, where MPSO_p and MPSO_d denote MPSO with the probabilistic scheme and the deterministic scheme, respectively. In MPSO_d, the LS operation is always executed for each species seed at each iteration, that is, the parameter p_{ls} is fixed to 1.0. The experimental results with respect to the convergence speed measurement are shown in Table 7.

From Table 7, it can be seen that MPSO_p always performs better than MPSO_d in terms of the convergence speed, which means that MPSO_p requires less evaluations to achieve all optima in the expected accuracy level than MPSO_d does. This happens because the LS operation in MPSO_p is executed by a probability p_{ls}, whose value can vary according to the measured ratio of valid LS operations n_v to total LS operations n_T. If the ratio $(\frac{n_v}{n_T})$ is greater than a given threshold, which means that LS is beneficial, the value of p_{ls} is increased; otherwise, LS is executed in the low level of efficiency and the value of p_{ls} is decreased. The experimental results show that this probability-based scheme is able to enhance the capability of LS effectively. In addition, MPSO_p outperforms LPSO on all test functions, which indicates that LS does help the algorithm locate the optimum more quickly.

Table 7
Experimental results with respect to the convergence speed of MPSO with the probabilistic and deterministic schemes on the test functions.

Funct.	LPSO	MPSO_p	MPSO_d
F1	1535	1324	1870
F2	1904	1828	2222
F5	1929	1679	5413
F6	27,643	14,472	31,541
F10	34,197	30,950	52,836

Table 8
Experimental results with respect to the accuracy of MSPO and peer algorithms on the test functions.

Funct.	MPSO_r		MPSO_non	
	Accuracy	Success rate (%)	Accuracy	Success rate (%)
F1	7.86E−16	100.00	1.07E−20	100.00
F2	1.57E−13	100.00	1.60E−10	100.00
F5	3.70E−14	100.00	1.03E−20	100.00
F6	5.08E−13	100.00	6.29E−04	93.33
F10	5.08E−13	100.00	6.67E−02	90.67

4.4.3. The effect of the triggered re-initialization scheme

From the basic experimental results, the triggered re-initialization scheme can help enhance the performance of MPSO on the test functions with numerous optima. The final set of experiments were carried out in order to examine the effect of this scheme upon the performance of MPSO on the test functions. The experimental results with respect to the accuracy and success rate measures are shown in Table 8, where MPSO_r and MSPO_non denote MPSO with and without the triggered re-initialization scheme, respectively.

From Table 8, it can be observed that MPSO_non cannot achieve all optima on F6 and F10, which indicates the validity of the proposed triggered re-initialization scheme. It is also noticeable that MPSO_non outperforms MPSO_r with a higher accuracy level on F1 and F5. This happens because the species construction method ensures that particles in the population distribute to each sub-region that only covers one single optimum, when the fitness landscape of a function has a few optima.

5. Conclusions

In this paper, a memetic PSO algorithm is proposed and experimentally investigated for multimodal optimization problems. In the proposed MPSO algorithm, a local version of PSO, where the species can be constructed according to the indices of particles in the population and used for locating multiple global and local optima in parallel, is hybridized with an adaptive LS method, which employs two different LS operations in a cooperative fashion. To further enhance the performance of MPSO for MMOPs with numerous optima, a triggered re-initialization scheme, where a converged species is re-initialized in order to enable the algorithm to search for new optima in the search space sequentially, is also integrated into the proposed MPSO algorithm.

In order to validate the proposed MPSO algorithm for MMOPs, experiments were carried out to investigate the performance of MPSO on a set of benchmark MMOPs in comparison with three state-of-the-art algorithms from the literature. From the experimental results, the following conclusions can be drawn on the test MMOPs.

First, a proper neighborhood topology structure enables the local PSO model to locate multiple optimal solutions in the search space simultaneously. For all test multimodal functions, MPSO always achieves all global and local optima with the success rate of 100%.

Second, LS is helpful to improve the performance of MPSO for MMOPs, but it is also problem-dependent. In the experiments, MPSO always performs much better than LPSO, while MPSO_1 outperforms MPSO_2 on F1, F2, and F5 and MPSO_2 performs better than MPSO_1 on F6 and F10. Therefore, the proposed adaptive LS method can help MPSO execute a robust local refinement since it employs multiple LS operators in a cooperative way.

Third, the triggered re-initialization scheme is beneficial to enhance the performance of MPSO in solving MMOPs with numerous optima. Comparing with the other three peer algorithms, only MPSO can successfully locate all optima on F9 and F10 in the expected accuracy level.

Generally speaking, the experimental results indicate that our proposed MPSO algorithm, which combines the above three schemes, seems a good optimizer for MMOPs.

For the future work, it is straightforward to examine the performance of the MPSO algorithm in comparison with other existing EAs or MAs for MMOPs. Another interesting research work is to extend the results in this paper to other EAs, e.g., differential evolution (DE) [29], and examine the performance of the obtained algorithms for MMOPs.

Acknowledgement

The authors would like to thank the guest editors and anonymous reviewers for their thoughtful suggestions and constructive comments. The work by Hongfeng Wang and Dingwei Wang was supported by National Nature Science Foundation of China (NSFC) under Grant 71021061, 70931001, 71001018 and 70801012, Fundamental Research Funds for the Central Universities under Grant N090404020 and N110404019 and Research Fund for the Doctoral Program of Higher Education of China under Grant 200801451053. The work by Ilkyeong Moon was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0025714). The work by Shengxiang Yang was supported by the Engineering and Physical Sciences Research Council (EPSRC) of UK under Grants EP/E060722/01 and EP/E060722/02.

References

[1] R. Brits, A.P. Engelbrecht, F. van den Bergh, Solving systems of unconstrained equations using particle swarm optimization, in: Proceedings of the 2002 IEEE International Conference on Systems, Man, Cybernetcis, 2002, pp. 102–107.
[2] R. Brits, A.P. Engelbrecht, F. van den Bergh, Locating multiple optima using particle swarm optimization, Applied Mathematics and Computation 189 (2007) 1859–1883.
[3] K.Y. Chan, T.S. Dillon, C.K. Kwong, Polynomial modeling for time-varying systems based on a particle swarm optimization algorithm, Information Sciences 181 (2011) 1623–1640.
[4] B. Bhanu, Y. Lin, Object detection in multi-modal images using genetic programming, Applied Soft Computation 4 (2004) 175–201.
[5] C.A.C. Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, Computer Methods in Applied Mechanics and Engineering 191 (11) (2002) 1245–1287.
[6] W. Chu, X. Gao, S. Sorooshian, Handling boundary constraints for particle swarm optimization in high-dimensional search space, Information Sciences (2010), doi:doi:10.1016/j.ins.2010.11.030.
[7] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolution Computation 6 (2) (2002) 182–197.
[8] J.E. Gallardo, C. Cotta, A.J. Ferandez, On the hybridization of memetic algorithms with branch-and-bound techniques, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 37 (1) (2007) 77–83.
[9] H. Huang, H. Qin, Z. Hao, A. Lim, Example-based learning particle swarm optimization for continuous optimization, Information Sciences (2010), doi:doi:10.1016/j.ins.2010.10.018.
[10] H. Ishibuchi, T. Yoshida, T. Murata, Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling, IEEE Transactions on Evolution Computation 7 (2) (2003) 204–223.
[11] T. Jansen, On the analysis of dynamic restart strategies for evolutionary algorithms, in: Proceedings of the 7th International Conference on Parallel Problem Solving from Nature, 2002, pp. 33–43.
[12] Y. Jin, J. Branke, Evolutionary optimization in uncertain environments-a survey, IEEE Transactions on Evolution Computation 9 (3) (2005) 303–317.
[13] J. Jezowski, R. Bochenek, G. Ziomek, Random search optimization approach for highly multi-modal nonlinear problems, Advances in Engineering Software 36 (2005) 504–517.
[14] Y. Juang, S. Tung, H. Chiu, Adaptive fuzzy particle swarm optimization for global optimization of multimodal functions. Information Sciences (2010), doi:doi:10.1016/j.ins.2010.11.025.
[15] J. Kennedy, The particle swarm: social adaptation of knowledge, in: Proceedings of 1997 IEEE International Conference on Evolution Computation, 1997, pp. 303–308.
[16] J. Kennedy, Stereotyping: improving particle swarm performance with cluster analysis, in: Proceedings of 2000 IEEE International Conference on Evolution Computation, 2000, pp. 1507-1512.
[17] J. Kennedy, R.C. Eberhart, Y. Shi, Swarm Intelligence, Morgan Kaufmann Publishers, San Francisco, CA, 2001.
[18] J. Li, M.E. Balazs, G. Parks, P.J. Clarkson, A species conserving genetic algorithm for multimodal function optimization, Evolutionary Computation 10 (2) (2002) 207–234.
[19] D. Liu, K.C. Tan, C.K. Goh, W.K. Ho, A multiobjective memetic algorithm based on particle swarm optimization, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 37 (1) (2007) 42–50.
[20] B. Liu, L. Wang, Y.H. Jin, An effective PSO-based memetic algorithm for flow shop scheduling, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 37 (1) (2007) 18–27.
[21] Q. Ling, G. Wu, Z. Yang, Q. Wang, Crowding clustering genetic algorithm for multimodal function optimization, Applied Soft Computing 8 (2008) 88–95.
[22] L. Liu, S. Yang, D. Wang, Force-imitated particle swarm optimization using the near-neighbor effect for locating multiple optima, Information Sciences (2010), doi:doi:10.1016/j.ins.2010.11.013.
[23] Z. Michalewicz, M. Schoenauer, Evolutionary algorithm for constrained parameter optimization problems, Evolutionary Computation 4 (1) (1996) 1–32.
[24] Y.-S. Ong, M. Lim, N. Zhu, K. Wong, Classification of adaptive memetic algorithms: a comparative study, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 36 (1) (2006) 141–152.
[25] D. Parrott, X. Li, Locating and tracking multiple dynamic optima by a particle swarm model using speciation, IEEE Transactions on Evolution Computation 10 (4) (2006) 440–458.
[26] Y.G. Petalas, K.E. Parsopoulos, M.N. Vrahatis, Memetic particle swarm optimization, Annals of Operations Research 156 (2007) 99–127.
[27] K.E. Parsopoulos, M.N. Vrahatis, On the computation of all global minimizers through particle swarm optimization, IEEE Transactions on Evolution Computation 8 (3) (2004) 211–224.
[28] J.E. Smith, Coevolving memetic algorithms: a review and progress report, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 37 (1) (2007) 6–17.
[29] R. Storn, K. Price, Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces, University of California, Berkeley, 2006.
[30] S. Salhi, N.M. Queen, A hybrid algorithm for identifying global and local minima when optimizing functions with many minima, European Journal of Operational Research 155 (2004) 51–67.
[31] O. Schutze, E. Talbi, C.C. Coello, L.V. Santana-Quintero, G.T. Pulido, A memetic PSO algorithm for scalar optimization problems, in: Proceedings of 2007 IEEE Swarm Intelligence Symposium, 2007, pp. 128–134.
[32] M.D. Toksar, Minimizing the multimodal functions with ant colony optimization approach, Expert Systems with Applications 36 (2009) 6030–6035.
[33] N. Tutkun, Optimization of multimodal continuous functions using a new crossover for the real-coded genetic algorithms, Expert Systems with Applications 36 (2009) 8172–8177.

[34] P.K. Tripathi, S. Bandyopadhyay, S.K. Pal, Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients, Information Sciences 177 (2007) 5033–5049.
[35] J. Tang, M.H. Lim, Y.-S. Ong, Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems, Soft Computing 11 (10) (2007) 957–971.
[36] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, Q. Tian, Self-adaptive learning based particle swarm optimization, Information Sciences (2010), doi:doi:10.1016/j.ins.2010.07.013.
[37] L. Xing, Y. Chen, H. Cai, An intelligent genetic algorithm designed for global optimization of multi-minima functions, Applied Mathematics and Computation 178 (2006) 355–371.
[38] S. Yang, Y.-S. Ong, Y. Jin (Eds.), Evolutionary Computation in Dynamic and Uncertain Environments, Springer-Verlag, Berlin Heidelberg, 2007.
[39] J. Zhang, D. Huanga, T.M. Lok, M.R. Lyu, A novel adaptive sequential niche technique for multimodal function optimization, Neurocomputing 69 (2006) 2396–2401.
[40] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Transactions on Evolution Computation 3 (4) (2006) 257–271.