Open access • Journal Article • DOI:10.1080/00207721.2011.605966

A memetic particle swarm optimisation algorithm for dynamic multi-modal optimisation problems — Source link

Hongfeng Wang, Shengxiang Yang, Wai Hung Ip, Dingwei Wang

Institutions: Northeastern University (China), Nanjing University of Information Science and Technology, Hong Kong Polytechnic University

Published on: 01 Jul 2012 - International Journal of Systems Science (Taylor & Francis Group)

Topics: Memetic algorithm, Particle swarm optimization, Local search (optimization) and Benchmark (computing)

A Memetic Particle Swarm Optimization Algorithm for Dynamic Multi-Modal Optimization Problems

Hongfeng Wang
hfwang@mail.neu.edu.cn
College of Information Science and Engineering, Northeastern University, Shenyang 110004, P. R. China
Shengxiang Yang
shengxiang.yang@brunel.ac.uk
Department of Information Systems and Computing, Brunel University, Uxbridge, Middlesex UB8 3PH, United Kingdom
W. H. Ip
mfwhip@inet.polyu.edu.hk
Department of Industrial and Systems Engineering, Hong Kong Polytechnic University, Hong Kong, P. R. China.
Dingwei Wang
dwwang@mail.neu.edu.cn
College of Information Science and Engineering, Northeastern University, Shenyang 110004, P. R. China

November 9, 2011

Abstract: Many real-world optimization problems are both dynamic and multi-modal, which require an optimization algorithm to not only find as many optima under a specific environment as possible, but also track their trajectory over dynamic environments. To address this requirement, this paper investigates a memetic computing approach based on particle swarm optimization for dynamic multi-modal optimization problems. Within the framework of the proposed algorithm, a new speciation method is employed to locate and track multiple peaks and an adaptive local search method is also hybridized to accelerate the exploitation of species generated by the speciation method. In addition, a memory-based re-initialization scheme is introduced into the proposed algorithm in order to further enhance its performance in dynamic multi-modal environments. Based on the moving peaks benchmark problems, experiments are carried out to investigate the performance of the proposed algorithm in comparison with several state-of-the-art algorithms taken from the literature. The experimental results show the efficiency of the proposed algorithm for dynamic multi-modal optimization problems.

Keywords: Memetic computing, Memetic algorithm, Particle swarm optimization, Dynamic multi-modal optimization problem, Speciation, Local search

1 Introduction

In the past decade, a class of new meta-heuristic methods, named memetic algorithms (MAs), which draw the inspiration from Darwinian principles of natural selection and Dawkins' notion of cultural evolution, have been widely used for solving many hard optimization problems, such as constraint optimization problems [7, 21], multi-objective optimization problems [17, 29, 42], and so on [22, 23]. MAs can usually be regarded as hybrid evolutionary algorithms (EAs) or extensions of EAs as a result of introducing local search (LS) methods into EAs as the separate process of individual learning for accelerating the search process. Obviously, such good balance between the local exploitation directed by LS methods and the global exploration directed by common EA operators is very helpful to enhance the capacity of MAs for solving complicated problems. This is also indicated in recent works on MAs where MAs are shown to achieve high-quality solutions more efficiently than their conventional counterparts on a lot of practical industry and engineering applications [15, 24, 25, 28].

Recently, dynamic optimization problems (DOPs) have gained more and more concern from the evolutionary computation community [2, 6, 10, 34, 35, 39, 40] since many real-world optimization problems

are always subject to dynamic environments, where the problem's objective function, decision variables, and/or constraints may change over time. For these DOPs, an optimization algorithm is always requested to track the changing optimum in the solution space as closely as possible, which is difficult to achieve for traditional EAs due to the convergence problem: once a population is converged, it cannot re-locate new optimal solutions quickly when an environmental change occurs. It is also noticeable that DOPs are often multi-modal, i.e., there exist multiple peaks in the fitness landscape. Obviously, these dynamic multi-modal optimization problems (DMMOPs) present more serious challenges to EAs since the goal would be to track as many as possible moving optima simultaneously, rather than only a single optimum.

In order to solve DMMOPs efficiently, researchers have developed a number of approaches into EAs, such as niches techniques, speciation methods, and multi-population strategies. These approaches mostly focus on how to distribute the individuals of the population into different search areas in the solution space in order to enable the algorithm to locate multiple optima. They seldom concern how to achieve these optima with a faster speed and higher accuracy. However, MAs own a distinct advantage: they can obtain very high-quality solutions quickly due to the introduction of the individual level learning mechanism. Therefore, it becomes a very interesting research topic to investigate dedicated memetic computing approaches for solving DMMOPs.

Inspired from the simulation of social behaviors of a group of fishes or birds, particle swarm optimization (PSO) has been used widely to solve many optimization problems as a robust technique in recent years. 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*). Here, how to choose *gbest* to guide the moving of each particle in the search space is a critical issue. Based on the learning approach of particles, PSO comes with two versions, i.e., the global version and the local version. In the global PSO version, all particles learn from the best particle in the 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 to locate on different optima in the solution space efficiently, which also leads to the application of PSO for solving multi-modal optimization problems (MMOPs) in both static and dynamic environments in recent years. Recently, PSO has been integrated into a memetic computing approach for optimization by Caponio *et al.* [8].

With the hybridization of PSO and LS, we propose a memetic computing approach, which is called MPSO, for DMMOPs in this paper. In the proposed algorithm, a new speciation method, where the particles are placed 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 method, where two different LS operators are employed in a cooperative fashion, is used to enhance the exploitation capacity of the MPSO algorithm. In addition, a memory-based re-initialization scheme is also introduced in order to further improve the performance of MPSO for DMMOPs. Extensive experiments are carried out on the Moving Peaks Benchmark (MPB) problem [3] in order to examine the major features of the proposed MPSO algorithm and to compare its performance with some peer algorithms taken from the literature.

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

2 Related work

Researches on DOPs [9, 19] or MMOPs [16] have gained a lot of concern from the evolutionary computation community in the last decade. However, these two fields are typically treated separately and a few works on EAs in both dynamic and multi-modal environments have just been reported in the very recent years. In this section, the related works on PSO in this research domain will be reviewed briefly. As for other EAs, the readers are referred to [18, 20, 31] for a comprehensive overview.

2.1 Principles of particle swarm optimization

PSO is a stochastic optimization algorithm where a population of individuals (particles) move through the search space [14]. The rules, which govern this movement, 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:

$$\vec{v}_i(t+1) = \omega\vec{v}_i(t) + c_1\xi(\vec{p}_i(t) - \vec{x}_i(t)) + c_2\eta(\vec{p}_{gi}(t) - \vec{x}_i(t)) \quad (1)$$

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

where $\vec{v}_i(t)$ and $\vec{x}_i(t)$ represent the current velocity and position of particle i at iteration t, respectively, $\vec{p}_i(t)$ is the position of the best solution (*pbest*) found so far by particle i, $\vec{p}_{gi}(t)$ is the position 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 uniform random numbers in the range $[0, 1]$.

Based on the approach of choosing *gbest*, PSO can be classified into two versions, i.e., the global version and the local version. In the global version of PSO algorithms, all particles in the population "share" the same *gbest* (the best fitness solution found so far by all particles). As a result, the population can converge quickly into one optimum in the search space. On the contrast, the local version of PSO algorithms 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 dynamic multi-modal environments more easily than the global PSO model.

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 too time-consuming because the distances between a particle and all other particles always require to be re-calculated when choosing 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, then particle i is neighbored to the particles with indices from $((i - r_s)\%N)$ to $((i + r_s)\%N)$.

However, there is a well-known problem in the simple local PSO model: it can not 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 DMMOPs, which is the major motivation of the research work in this paper.

2.2 PSOs for MMOPs

Over the past decade, MMOPs have been studied by many EA researchers and a number of strategies have been proposed, such as the clustering method, the niches technique, and so on. However, it is noticeable that most works focused on genetic algorithms (GAs) [30, 37], while PSO was considered for solving MMOPs, especially dynamic MMOPs (DMMOPs), only in very recent years.

Kennedy [13] initially reported his works on PSO for the static MMOPs using the clustering method, where a k-means clustering algorithm was used to identify the centers of different clusters of particles in the population, and then these cluster centers are used to substitute the personal bests (*pbest*) or neighborhood bests (*gbest*) for each particle. Brits *et al.* [4] introduced a *nbest* PSO for MMOPs in static environments, 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 earlier works lies in 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 niches technique is a common strategy for EAs in static multi-modal optimization environments. Brits *et al.* [5] introduced the niche technique into the PSO algorithm for static MMOPs, where multiple sub-swarms can be created by splitting a particle and its closest neighbor from a main swarm if that particle's fitness shows very little change over a number of iterations, and are used to locate multiple optima in parallel in the search space. Zhang *et al.* [41] proposed an adaptive sequential niche PSO (ASNPSO) algorithm for static MMOPs. In ASNPSO, multiple sub-swarms are created sequentially to locate multiple optimal solutions. In order to avoid all sub-swarms from converging to one or several certain optimal solutions, the particles in a sub-swarm that runs later need to modify their fitness based on a penalty function if they are determined to reside in the same peak that has been found by a previous sub-swarm that runs earlier. A hill valley function is designed to determine whether or not two points of the search space belong to a same peak of the multi-modal function through calculating the fitness of several interior sample points between these two points.

The idea behind the above PSO algorithms for the static MMOPs is to dispatch the particles in the population into different promising regions in the solution space and then construct multiple sub-swarms to exploit the corresponding search areas. Obviously, these algorithms lack the capacity of adapting well to the environmental dynamics because once the sub-swarms have been converged, they cannot re-achieve the new optimum quickly after an environmental change occurs. Furthermore, it is also necessary in PSO to maintain the capacity of exploring new peaks for DMMOPs, which was seldom considered for PSO algorithms in static environments.

Recently, PSO for DMMOPs has begun to gain more concern due to the importance and universality in the real-world applications. Parrott and Li [26] 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 species is also designed for maintaining the algorithm's capacity of exploring new peaks. Janson and Middendorf [11] proposed a hierarchical PSO, which can maintain a hierarchy of particles that are partitioned into several sub-swarms for a limited number of generations after an environmental change occurs. Blackwell and Branke [1] adopted a set of interacting swarms to track multiple optima simultaneously. Two special operators, exclusion and anti-convergence, can prevent multiple swarms from converging to the same peak and keep the exploration capacity for a new peak in the search space, respectively. In addition, the charged particles or quantum particles are used to encourage the diversity in each sub-swarm. In the work of Yang and Li [38], a hierarchical clustering method is used to create automatically multiple sub-swarms depending on the distribution of particles in the initial population. In the following iterations, an overlapping check scheme between each pair of sub-swarms is used to prevent multiple sub-swarms from covering the same optimum, and an overcrowding check scheme is also performed on each sub-swarm to avoid too many particles searching on a single peak and hence save the computing resources. Once an environmental change is detected, the clustering method is applied for the new initial population, where all achieved optima are retrieved through replacing the worst initialized particles. In addition, a faster LS method was also introduced to search optimal solutions in a promising sub-region found by the clustering method, which has the same purpose as in this paper.

3 The proposed algorithm

3.1 The neighborhood topology structure

Based on the above description, it is easy to understand that the local PSO model can adapt better to the dynamic multi-modal environments than the global PSO model since particles in the local model can learn from their own *gbest*s, which is helpful to the algorithm for tracking multiple changing optimum simultaneously. Obviously, one key question in the local PSO model is how to choose *gbest* for each individual in the population, i.e. how to design the neighborhood topology structure.

Recently, a speciation method [26] was introduced into PSO for DMMOPs. It is noticeable that the original species method always requires many evaluations via the Euclidean distance during the

```
Procedure Species Formation Algorithm:
begin
  S := ∅;
  for each x⃗ in P do
    if x⃗ does not belong to a "full" speciesthen
      set the status of x⃗ as unprocessed;
    endif
  endfor
  while (there are unprocessed particles in P) do
    get the best unprocessed particle x⃗_k ∈ P;
    set seed := true;
    for each s⃗ ∈ S do
      if d(x⃗_k, s⃗) < r_0 then
        re-initialize x⃗_k;
        set seed := false;
        break;
      endif
    endfor
    if (seed) then
      S := S ∪ {x⃗_k};
      for each x⃗ with index in [k − r_s, k + r_s] do
        if x⃗ is unprocessed then
          set x⃗_k as x⃗'s pbest;
          set x⃗'s status as processed;
        endif
      endfor
    endif
  endwhile
end
Denotations:
  S: a list of selected species seeds;
  P: the current population;
  d(x⃗, y⃗): the Euclidean distance between two particles x⃗ and y⃗;
  r_s: a parameter of controlling the size of a species;
  r_0: a parameter of controlling the search region of a species.
```

Figure 1: Pseudo-code for the algorithm of forming species.

species identification course since a species is defined as a group of particles according to the similarity metric in the Euclidean distance space. In our previous work [36], we employed a special PSO model, where all particles are located into a ring-shape topology structure, to decide the neighbors of each particle. The radius of neighborhood could be measured by the population index distance between two particles. Obviously, this neighborhood measurement has a less time complexity than the neighborhood measurement based on the Euclidean distance. However, it is very regretful that this simple scheme may eventually lead to the population converging into few optima or even only one optimum due to the interaction among the particles, which is not expected when solving DMMOPs. The similar results can be validated in [36], where a self-organized random immigrants scheme is employed to improve the capacity of PSO with this ring topology structure in dynamic environments.

Inspired by the aforementioned works, a new speciation method is proposed in this paper, where the particles adaptively form multiple different species based on their population indices in a ring-shape topology structure. The pseudo-code on how to form the species can be summarized in Figure 1, where

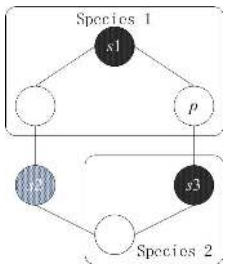

Figure 2: An example of how to form the species in a population of six particles ($r_s = 1$).

a "full" species is defined as a species with the maximal allowable number of particles.

The speciation forming algorithm is performed at each iteration. The species seed set S is initially set to $\emptyset$ and then the status of each particle in the population P is set as unprocessed. The status of a particle would be used to judge whether this particle will participate in the following operations or not. The best unprocessed particle $\vec{x}_k$ is firstly selected from P to check whether it will become a new species seed or not. If $\vec{x}_k$ falls within the radius r_0 from any existed seed $\vec{s}$ in S, it can not become a species seed and only be initialized straightly, that is, its position and velocity would be re-generated randomly. Otherwise, $\vec{x}_k$ will be chosen as a new seed and added to S. Then, a new species will be constructed by all unprocessed particles whose indices are within the range $[k - r_s, k + r_s]$. This course will continue until all particles in the population have been processed.

Figure 2 provides an example to illustrate how the above algorithm works. In this case, we suppose that the population has six particles, of which $\vec{s}_1$, $\vec{s}_2$ and $\vec{s}_3$ are the first, second, and third fittest particles, respectively. We also suppose that the Euclidean distance between $\vec{s}_1$ and $\vec{s}_2$ is lower than the threshold r_0, while the Euclidean distance between $\vec{s}_1$ and $\vec{s}_3$ is larger than r_0. Based on our proposed speciation method, the first species can be formed by $\vec{s}_1$ and its two neighbor particles whose indices are closest to $\vec{s}_1$. Then, $\vec{s}_2$ can not be used to form one species since it may locate in the same peak as $\vec{s}_1$. Finally, another species dominated by $\vec{s}_3$ can be formed since the Euclidean distance between $\vec{s}_3$ and $\vec{s}_1$ is large. Note also that the particle $\vec{p}$ in Figure 2 only belongs to the species dominated by $\vec{s}_1$, 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 their radii overlap.

From the above description, it can be seen that the particles in the population will form automatically species of different sizes using the above speciation method. Obviously, the larger the size of a species, the more easily the species achieves one optimum located in its search region. In order to accelerate the speed of one species achieving one optimum, the status of all particles in a "full" species is always set to "processed" during the running course of the algorithm. This means that a species does not participate in the following species forming operation once it has been created as a "full" one. In Figure 2, given $r_s = 1$, the maximal allowable number of particles in a species is 3. Hence, species 1 is a "full" species, while species 2 is not a "full" species. Note that at least one "full" species would be created using the proposed speciation method.

The species-based method aims at forming multiple species to explore different search regions of the solution space in parallel. The difference between the speciation scheme by Parrott and Li [26] and our proposed speciation method lies in how to identify the members of one species once its seed is identified. In Parrott and Li's speciation scheme, one species is composed of all particles that fall within a given Euclidean distance from its species seed, while one species in our speciation method can include a segment of particles surrounding its species seed in a ring-topology structure. Of course, it is still necessary to check whether one species seed falls in the search area of any existing species in our proposed method in order to avoid two different species exploring the same peak in the solution space. It is obvious that our proposed speciation method will consume less computational time. In the example of Figure 2, the number of Euclidean distance calculations is 3 using our proposed method, while it is at least 5 using the speciation method in SPSO [26]. Since the search of one species is mainly determined by its seed, the two speciation methods would have the same effect on the division of the search space, which is also

Procedure $NCLS(\vec{x})$:
begin
  $\vec{x}' := \vec{x}$;
  initialize $\vec{v}'$ within a small range;
  $\vec{p}' = N(\vec{p}, \sigma)$;
  for $i := 1$ **to** n_{ls} **do**
    $\vec{x}' := \vec{x} + \omega\vec{v}' + c_1\xi(\vec{p}' - \vec{x})$;
    if $f(\vec{x}') > f(\vec{x})$ **then** $\vec{x} := \vec{x}'$;
    if $f(\vec{x}') > f(\vec{p})$ **then**
      $\vec{p} := \vec{x}'$;
      $\vec{p}' = N(\vec{p}, \sigma)$;
    endif
  endfor
end;
Denotations:
  $\vec{x}$: the selected particle for local improvement
  $\vec{v}'$: the velocity of each LS move
  $\vec{p}$: *pbest* of a selected particle $\vec{x}$
  $\vec{p}'$: the stochastic position based on $\vec{p}$, characterized by a normal distribution vector $N(\vec{p}, \sigma)$
  n_{ls}: the step size of LS, i.e., the number of moves

Figure 3: Pseudo-code for the NCLS operator.

validated in the experimental study later on in this paper.

3.2 Local search

In MAs [32, 33], EA operators are used for rough global exploration and LS methods are used for refining local exploitation. So, LS plays a very important role in enhancing MAs' exploitation capacity. In the context of PSO, the LS procedure can be regarded as a particle making one iterative biased or random local move, while the moving from the current solution to a candidate solution will be accepted if the candidate solution has a better fitness. To design robust LS operators, several key issues need to be considered.

The first issue concerns which LS operations should be used by the particles that are selected for local refinement. Here, two different LS operators are considered in our proposed algorithm, which are described as follows.

1) Non-deterministic Cognition-based Local Search ($NCLS$): This LS operator is designed according to a special PSO model. Kennedy [12] introduced two special models of PSO, 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 a particle's exploitation to its own best *pbest*. In order to enable the particle track a moving optimum well in dynamic environments, an NCLS method is designed into our proposed algorithm, and its pseudo-code is as shown in Figure 3, where a maximization problem is assumed.

From the pseudo-code, it can be seen that σ is an important parameter to affect the performance of the $NCLS$ operator because the move of a particle is directed by the stochastic position of *pbest*. The smaller the value of σ, the closer the position of the generated stochastic *pbest* to its original position. This means that the particle can execute one biased move toward a smaller area around *pbest*. When $\sigma = 0$, the particle's move is decided by its *pbest* absolutely as a result that the *non-deterministic cognition-only model* returns to the *cognition-only model*. Moreover, the initialization of $\vec{v}'$ (see Figure 3) is also confined

Procedure $RWDE(\vec{x})$:
begin
  for $i := 1$ **to** n_{ls} **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 $\lambda := \lambda/2$;
    if $f(\vec{x}') > f(\vec{p})$ **then** $\vec{p} := \vec{x}'$;
  endfor
end
Denotations:
  λ: a prescribed scalar step-length
  Other parameters are the same as those in Figure 3

Figure 4: Pseudo-code for the $RWDE$ operator.

within a small range in order to reduce the influence of the random factor.

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 [27]. Rather than directing the moving of a particle by its stochastic *pbest* in the $NCLS$ 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 Figure 4.

From the pseudo-codes in Figure 3 and Figure 4, it can be seen that the two LS operators adopt different moving strategies, respectively. The $NCLS$ operator enables a particle to search around its *pbest* position stochastically, 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.

The second issue of designing LS operators concerns which individuals in the population should be improved by an LS operator. In many researches on MAs, the LS operation is applied to each newly generated individual in order to obtain a higher-quality solution, which seems to be too costly and infeasible for MAs in dynamic environments. Hence, it becomes a good choice that any species seed should be executed LS operation if the species could cover one peak in the solution space since DMMOPs are considered in this paper. However, it is difficult to confirm whether one species covers one peak. In our proposed algorithm, the LS operation is only applied to the seed of "full" species considering that the "full" species can cover one peak more easily than the "non-full" ones.

The third issue concerns the computational time cost of LS operators. When designing LS operators in MAs for DOPs, their computational cost has to be considered since the environment may change over time. To decrease the number of evaluations occupied by useless LS operations as much as possible, two strategies are designed in this paper since the number of fitness evaluations will be used to measure the computational time of algorithms. One strategy is that LS is applied to each seed of "full" species by a probability p_{ls}, of which the value can be calculated according to the convergence state of the species. Here, the convergence state of one species can be measured by the following formula.

$$div(s) = \frac{1}{|s|}\sum_{i=1}^{|s|} d(s_i, s_{center}). \tag{3}$$

where $div(s)$ denotes the convergence index of species s, s_i denotes the i-th member of s, s_{center} and $|s|$ denote the center position and size of s, respectively. From this formula, it is easy to see that the species is approaching convergence if $div(s) \to 0$. Considering that the LS operation can help to improve the seed's quality of a diverse species more efficiently than that of a converged species, a robust scheme is used to calculate the value of p_{ls}. In this scheme, p_{ls} is initially set to p_0 and, then, adjusted adaptively

Procedure Adaptive LS method:
begin
  if p_{ls} is not initialized **then**
    set p_{ls}=p_0;
  else
    calculate p_{ls} according to Eq. (4);
  if $rand() < p_{ls}$ **then**
    if n_{ls} is not initialized **then**
      set n_{ls}=n_0;
    else
      calculate n_{ls} according to Eq. (5) and Eq. (6);
    for $i := 1$ **to** n_{ls} **do**
      $\vec{x}' = \vec{x}$;
      if $d(\vec{x}, \vec{p}) < r_1$ **then** $RWDE(\vec{x}')$;
      else $NCLS(\vec{x}')$;
      if $f(\vec{x}') > f(\vec{x})$ **then** $\vec{x} := \vec{x}'$;
      if $f(\vec{x}') > f(\vec{p})$ **then** $\vec{p} := \vec{x}'$;
    endfor
end
Denotations:
  $rand()$: a random generated number in (0,1)
  r_1: a very small positive number
  Other parameters are the same as those for $NCLS$

Figure 5: Pseudo-code for the adaptive LS method.

according to the convergence state of a species, which is shown as follows:

$$p_{ls}(s_t) = \begin{cases} p_{ls}(s_{t-1}), div(s_t) = div(s_{t-1}) \\ min\{p_{ls}^{max}, p_{ls}(s_{t-1})/\alpha\}, div(s_t) > div(s_{t-1}) \\ max\{p_{ls}^{min}, p_{ls}(s_{t-1}) \cdot \alpha\}, div(s_t) < div(s_{t-1}) \end{cases} \tag{4}$$

where s_t and s_{t-1} denote a "full" species which is executing an LS operation and the species which has just executed an LS operation, respectively, p_{ls}^{max} and p_{ls}^{min} denote the maximal and minimal value of p_{ls} ($p_{ls}^{max} = 1.0$ and $p_{ls}^{min} = 0.1$ in the later experiments), respectively, and $\alpha \in (0, 1)$ is a preset constant. Obviously, the purpose of this scheme is to encourage the execution of LS operations upon the seed of a diverse species.

Another strategy is to adjust the step size n_{ls} of local moving, of which the initial value is set to n_0, based on the effect of LS. Let $n_v(t-1)$ and $n_T(t-1)$ denote the actual number of valid local movements and total local movements carried out during iteration $t-1$, respectively. A valid local moving means that a local moving that improves the quality of a particle. Then, the value of n_{ls} is calculated as follows:

$$sign(t) = \begin{cases} 1, \frac{n_v(t-1)}{n_T(t-1)} < \delta \\ 0, \frac{n_v(t-1)}{n_T(t-1)} = \delta \\ -1, \frac{n_v(t-1)}{n_T(t-1)} > \delta \end{cases} \tag{5}$$

$$n_{ls}(t) = min\{n_{ls}^{max}, max\{\beta^{sign(t)} \cdot n_{ls}(t-1), n_{ls}^{min}\}\} \tag{6}$$

where $\delta \in (0, 1)$ is a preset constant that acts as the threshold of changing n_{ls}: decreasing, increasing, or neither. $\beta \in (0, 1)$ controls the decreasing or increasing speed of n_{ls}, and n_{ls}^{max} and n_{ls}^{min} are predefined maximal and minimal value of n_{ls} ($n_{ls}^{max} = 5$ and $n_{ls}^{min} = 1$ in this paper), respectively.

Based on the above discussions, an adaptive LS method, where $NCLS$ 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 Figure 5. From Figure 5, the probability p_{ls} firstly requires to be updated when the adaptive LS method is applied to an individual. Of course, the value of p_{ls} would be set to p_0 if it is not initialized. If a generated random value $rand() \in (0, 1)$ is less than p_{ls}, the individual that is selected for local improvement will check the distance between its *pbest* and itself after the local moving step n_{ls} is determined. If the distance is less than a preset constant, the $RWDE$ operation will be applied on the selected individual; otherwise, the $NCLS$ operation will be executed on it. It is easy to understand that $NCLS$ may become a random search when the particle approaches too closely to its *pbest* according to the pseudo-code in Figure 3.

3.3 Memory-based re-initialization

Based on the proposed speciation method, the population can be divided into a number of species to locate multiple optima in parallel. Obviously, the number of species, which can be determined by 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 as many as possible potential solutions. However, this expectation can not be achieved since the number of optima is always unknown in advance, while the algorithm parameters need to be predefined. In order to address this problem, a memory-based re-initialization method is proposed and discussed in this section.

In the memory-based re-initialization scheme, each species will be checked to see whether it has converged into one optimum at each iteration. If the species converges into an optimum, its species seed will be copied into a memory pool M as an achieved optimum and then all the members of the species will be re-initialized randomly. Here, the convergence index $div(s)$ described in the previous section is used to judge whether a species s has converged into an optimum or not. If the value of $div(s)$ is lower than a certain threshold r_2, the species s will be re-initialized and the species seed is copied into M.

However, one problem that follows when this memory-based scheme is integrated into our MPSO algorithm is that the convergence condition may mistake a non-optimal solution as an optimum in some cases. In the proposed MPSO model, the 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 s, the value of $div(s)$ is always equal to 0. It is obvious that such species may not achieve one optimal solution. Therefore, only the "full" species will be checked whether to be re-initialized at each iteration. Of course, all particles in a converged "full" species are reset as "unprocessed" once it is re-initialized.

To avoid a species from re-exploiting one of the achieved peaks in the memory pool M, the species will evaluate the Euclidean distance between the positions of its seed and each achieved solution in M after accomplishing its update. If the distance is less than r_0, the species will be re-initialized because it may re-explore the search region dominated by one achieved solution. In addition, all solutions in M should be retrieved into the population when one environmental change is detected, which is also a common strategy used in many researches on EAs for DOPs.

In the proposed algorithm, we use the best particle over the whole population as the monitoring particle to detect environmental changes. Before updating the best particle, we re-evaluate its fitness at each iteration. If its fitness changes, it indicates that an environmental change occurs. Once an environmental change is detected, all particles are re-evaluated and all achieved solutions in M replace the same number of worst species seeds in the current population if these solutions have better fitness after being re-evaluated.

In summary, in this paper, an adaptive LS method and a memory-based re-initialization scheme are both integrated into the local PSO model with the neighborhood topology structure described in Section 3.1 for solving DMMOPs. The proposed MPSO algorithm for DMMOPs is as shown in Figure 6.

4 Experimental study

In this section, experiments are carried out in order to evaluate the performance of our proposed MPSO algorithm and to study its main features based on the MPB problem [3]. Two groups of experiments are carried out. In the first group of experiments, we study the influence of the proposed two major operators in this paper, that is, the adaptive LS operator and the new speciation method, upon the performance of MPSO for DMMOPs. In the second group of experiments, we examine the performance of MPSO for

Procedure The Proposed MPSO Algorithm:
begin
  generate an initial population P randomly;
  repeat
    determine the species using the algorithm in Figure 1;
    check the convergence state of each "full" species and update memory set M;
    execute LS operation in Figure 5 for each seed of "full" species;
    update each particle in P according to Eqs. (1) and (2) respectively;
    re-initialize each "full" species falling within the search region of any achieved solution in M;
    retrieve all achieved solutions in M into P if an environmental change is detected;
  until a termination condition is met
end

Figure 6: Pseudo-code for the proposed MPSO algorithm.

DMMOPs in comparison with several peer algorithms taken from the literature. The involved algorithms include Blackwell and Branke's mCPSO and mQPSO [1], Parrott and Li's SPSO [26], the self-organizing random immigrants memetic PSO (SOMPSO) algorithm, which was denoted as LPSO-SOFCMA in [36], and the clustering PSO (CPSO) algorithm in [38]. All results of peer algorithms shown in this paper are provided in the papers where they were proposed.

4.1 Experimental Setup

1) *Dynamic test environments*: The MPB problem [3] has been widely used in the literature as the test DOP. It consists of a number of peaks, of changing heights, widths, and locations in a random direction. In a n-dimensional real space, the MPB problem with m peaks can be defined as follows:

$$F(\vec{x},t) = \max_{i=1,\ldots,m} \frac{H_i(t)}{1 + W_i(t)\sum_{j=1}^{n}(x_j(t) - X_{ij}(t))^2}, \tag{7}$$

where $H_i(t)$ and $W_i(t)$ are the height and width of peak i at time t, respectively, and $X_{ij}(t)$ is the j-th element of the location of peak i at time t. Each peak can independently change its height and width and move its location around in the search space. In detail, the height and width of each peak are changed by adding a random Gaussian variable and the location of each peak is moved by a shift vector $\vec{v_i}$ of a fixed length s, which can be described as follows:

$$\begin{cases} \delta \in N(0,1) \\ H_i(t) = H_i(t-1) + height\_severity \cdot \delta \\ W_i(t) = W_i(t-1) + width\_severity \cdot \delta \\ \vec{X}_i(t) = \vec{X}_i(t-1) + \vec{v}_i(t), \end{cases} \tag{8}$$

$$\vec{v}_i(t) = \frac{s}{|\vec{r} + \vec{v}_i(t-1)|}((1-\lambda)\vec{r} + \lambda\vec{v}_i(t-1)), \tag{9}$$

where the shift vector $\vec{v}_i(t)$ is a linear combination of a random vector $\vec{r}$ and the previous shift vector $\vec{v}_i(t-1)$ and is normalized to length s. The random vector $\vec{r}$ is created by drawing random numbers for each dimension and normalizing its length to s.

The default settings and definition of the benchmark used in the experiments of this paper correspond to Scenario 2 as specified on the benchmark website [3] and can be found in Table 1, which are the same as in all the involved algorithms. In Table 1, the term "change frequency (U)" means that the environment changes every U fitness evaluations, S denotes the range of allele values, and I denotes the initial height for all peaks. The dynamism of changes is described as follows. The height of peaks is shifted randomly in the range $H = [30.0, 70.0]$ and the width of peaks is shifted randomly in the range $W = [1.0, 12.0]$.

Table 1: Default setting for the MPB problem

parameter	value
no of peaks (m)	10
no of dimensions (n)	5
change frequency/period (U)	5000
shift length (s)	1.0
correlation coefficient (λ)	0
$height\_severity$	7.0
$width\_severity$	1.0
H	[30.0,70.0]
W	[1.0,12.0]
S	[0.0,100.0]
I	50.0

2) *Experimental Setting*: In MPSO, the population size was set to 50, the cognitive and social learning factors c_1 and c_2 were both set to 1.4962 and the inertia weight ω was set to 0.72984. The velocity of a particle is always confined within the range $[-V_{MAX}, +V_{MAX}]$, i.e., between the lower and upper bounds of the range of variables (here $V_{MAX} = 100$). The parameters r_s and r_0 in the proposed speciation method were set to 2 and 10.0, the parameters $r_1 = 0.01$, $p_0 = p_{ls}^{max}$, $n_0 = n_{ls}^{max}$, $\alpha = 0.5$, $\beta = 0.2$, $\delta = 0.5$ in the proposed adaptive LS method and the parameter r_2 was set to 0.0001 in the proposed memory-based scheme.

For each experiment of an algorithm on a test problem, 50 independent runs were executed with the same set of random seeds. For each run of an algorithm on a DOP, 100 environmental changes were allowed, which result in $K \times U = 5 \times 10^5$ fitness evaluations. The performance of an algorithm can be measured by an offline error, which is defined as follows:

$$\mu = \frac{1}{K}\sum_{k=1}^{K}(h_k - f_k), \tag{10}$$

where f_k is the best solution obtained by an algorithm just before the k-th environmental change, h_k is the optimum value of the k-th environment, μ is the average of all differences between h_k and f_k over the environmental changes, and K is the total number of environments.

4.2 Experiments on Investigating the Major Features of MPSO

In order to investigate the effect of designed major operations upon the performance of MPSO, the following two primary experiments are carried out in this section.

1) *Testing the effect of LS operators*: In our proposed MPSO algorithm, an adaptive LS method, which adopts two different strategies for local improvement based on a probability, is designed to refine the quality of seeds of "full" species to enable the algorithm to achieve the optimum quickly. We study the effect of different operations in this LS method in order to examine their validity in this section.

In the first set of experiments, we investigate the performance of MPSO with different LS operators on the static period of the MPB problem, which means the maximum allowable evaluations was set to 5000 only. The reason why the algorithms were executed only on the static problem is that the function of LS is the same in fact in both static and dynamic environments. For the convenience of description, MPSO_0, MPSO_1, and MPSO_2 are used to denote MPSO with the proposed adaptive LS operator, MPSO with only the NCLS operator, and MPSO with only the RWDE operator, respectively, while LPSO denote MPSO without any LS operator. In all investigated algorithms, the execution probability and step size of LS are adjusted adaptively as described in Section 3. The experimental results are shown in Figure 7, where the data were averaged over 50 runs with the same set of random seeds.

From Figure 7, it can be observed easily that MPSO algorithms perform better than LPSO, which shows that a proper LS operator is helpful in improving the performance of the PSO algorithm. Of

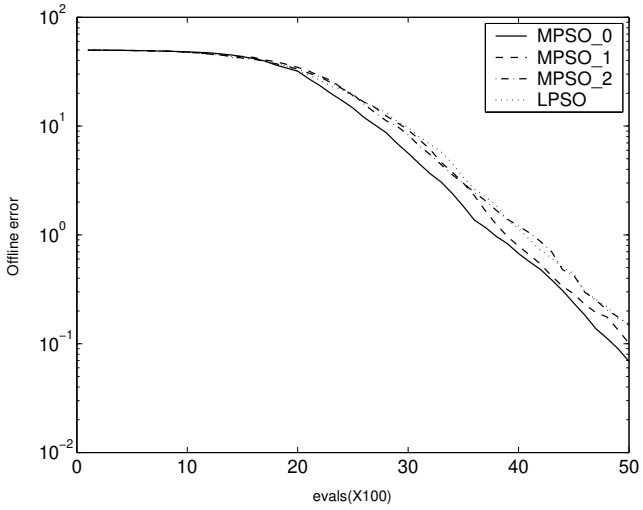

Figure 7: Experimental results of MPSOs with or without different LS operators on the stationary test problem.

course, LS is not always effective. For example, MPSO_2 exhibits almost the same performance as LPSO during the late running period. Obviously, the good performance of MPSO_0 over MPSO_1 and MPSO_2 indicates that the proposed adaptive LS method can execute more robust and efficient local improvement.

Another feature in our proposed LS method is that the execution probability and step size of LS can be adjusted adaptively in order to decrease the number of useless operations given that the LS operation may consume extra computational time in terms of fitness evaluations. In the following experiments, we evaluate the influence of designed schemes on how to calculate the execution probability and step size upon the proposed LS method. In particular, four different algorithms were tested on the static MPB problem, which are described as follows. MPSO_p0 and MPSO_p1 denote the probabilistic MPSO algorithms with an adaptive LS step size and with a fixed LS step size, respectively, while MPSO_d0 and MPSO_d1 denote the deterministic MPSO algorithms with the adaptive LS step size and with a fixed LS step size, respectively. In the deterministic MPSOs, the LS operation is always executed for each selected particle, that is, the probability p_{ls} is fixed to 1.0. The experimental results are shown in Figure 8.

From Figure 8, the following results can be observed. Firstly, the probabilistic scheme can help to enhance the performance of the LS technique. In the experiments, MPSO_p0 greatly outperforms MPSO_d0 and MPSO_p1 outperforms MPSO_d1. The reason lies in that LS operations always consume additional evaluations, while too many useless LS operations are not beneficial to improve the performance of the algorithm. In the early search period of the algorithm (e.g., before 2000 evaluations), most LS operations are effective since the quality of individuals is low. However, the number of useless LS operations can increase acutely once the population converges into the high performance area (after 2000 evaluations). Obviously, it can be observed from the results in Figure 8 that the probabilistic scheme is helpful to reduce the influence of these useless LS operations. Secondly, the scheme of adapting the LS step size can also decrease the computational cost of useless LS operations with a certain degree, which can be shown from the experimental results that MPSO_p0 performs better than MPSO_p1 and MPSO_d0 performs better than MPSO_d1. However, it is also noticeable that the probabilistic scheme is more effective than the varying step size scheme in improving the effect of LS.

2) *Examining the validity of the speciation method*: The working principle of the speciation mechanism is to divide the particles in the population into multiple independent species to achieve different peaks in the solution space simultaneously, which can help the algorithm maintain sufficient population diversity in dynamic multi-modal environments. In this paper, we propose a new speciation method which can construct different species according to the population indices of particles in a ring-shape topology structure. In order to examine the validity of this speciation method in keeping the population diversity, we compare the performance of MPSO with three peer algorithms, including SPSO, SOMPSO, and a simple

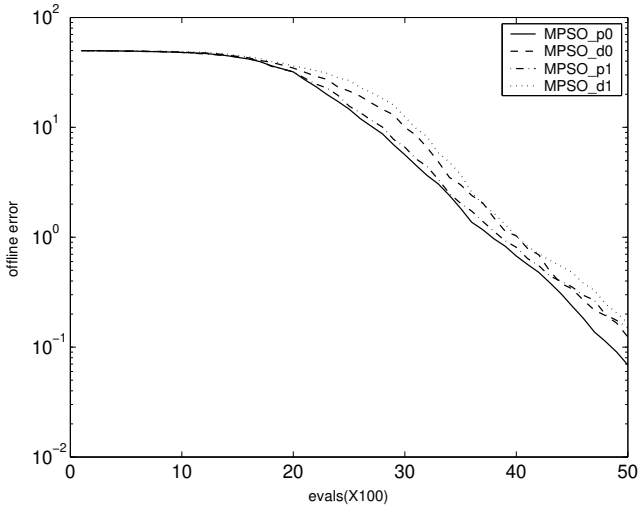

Figure 8: Experimental results of MPSOs with different execution schemes of LS on the stationary test problem.

Table 2: The number of peaks found by MPSO, SPSO, and sPSO on the MPB problems with different number of peaks

peaks	MPSO	SPSO	SOMPSO	sPSO
5	4.93	4.86	4.13	2.09
10	9.70	8.77	6.34	2.72
20	15.45	13.93	7.45	3.45
50	15.98	19.66	9.15	4.57
100	16.43	22.15	10.34	4.91

local PSO algorithm (denoted as sPSO) on the MPB problems with different settings. In SPSO, multiple species, which are constructed in the Euclidean distance space, are used to search different peaks in parallel, while the last two algorithms both utilize a ring topology with the neighbor radius set to 1. The difference between SOMPSO and sPSO lies in that SOMPSO employs a self-organized random immigrants scheme in order to improve its capacity of exploring the new search space. In addition, LS operators were also used to enhance the exploitation capacity of SOMPSO.

We investigate the scaling capacity of the speciation method with respect to the number of peaks in the MPB problem, considering that increasing the number of peaks makes it harder for algorithms to locate the optima. In particular, MPSO, SPSO, SOMPSO, and sPSO are executed on the MPB problems with the shift length s set to 1.0 and the number of peaks set to 5, 10, 20, 50, and 100, respectively. The experimental results with respect to the number of peaks found are shown in Table 2. If a peak falls within a certain radius (here the value was set to 0.5) of one particle in the population, it is considered to be found by an algorithm. Strictly speaking, we cannot assume that a peak has been found just because the Euclidean distance between it and one particle in the population is less than a given radius. We just use this simple approximate measure to examine how many peaks can be located by the species generated by the speciation method in our experiments.

From Table 2, it can be seen that MPSO and SPSO always achieve much more peaks than SOMPSO dose. For example, the average number of peaks found by MPSO and SPSO is 15.45 and 13.93, respectively, while SOMPSO only achieves on the average 7.45 peaks on the MPB problem with 20 peaks. These results indicate that the investigated speciation methods in MPSO and SPSO can help them to maintain better population diversity than the self-organized random immigrants scheme in SOMPSO.

Table 3: The number of peaks found by MPSO, SPSO, SOMPSO, and sPSO on the MPB problems with 10 peaks and different shift severities

s	MPSO	SPSO	SOMPSO	sPSO
0.5	9.72	9.10	6.72	2.74
1.0	9.70	8.77	6.34	2.72
2.0	9.61	9.10	6.27	2.72
5.0	8.93	7.57	5.91	2.68

Table 4: The offline error of MPSO and the peer algorithms on the MPB problems with different number of peaks

peaks	MPSO	SPSO	SOMPSO	CPSO	mCPSO	mQPSO
5	0.56±0.09	1.50±1.15	0.58±0.08	0.72±0.30	2.11±0.10	1.80±0.08
10	0.71±0.20	2.28±1.10	2.31±1.47	1.06±0.24	2.09±0.09	1.83±0.06
20	1.11±0.56	3.46±1.28	4.37±1.33	1.59±0.22	2.71±0.23	2.50±0.12
50	1.58±0.69	5.27±1.63	5.74±1.25	1.54±0.12	2.83±0.31	2.61±0.15

It can also be seen that MPSO can achieve the same number of peaks as SPSO does when the number of peaks is not huge. Obviously, such experimental result confirms our prediction that the species can be constructed based on the indices of particles in the population. However, it is noticeable that SPSO can achieve more peaks than MPSO when there are numerous peaks in the search space. This happens because that the LS operation in MPSO requires some additional computational cost. In addition, sPSO can only achieve few peaks on all test problems, which shows that a simple local PSO model cannot improve the population diversity of the PSO algorithm effectively.

We further carry out the following set of experiments on the MPB problems with different settings of the shift length s in order to examine the capacity of the speciation method in adapting to different environmental dynamics. In particular, the above four algorithms are executed on the test problems with 10 peaks and the value of s was set to 0.5, 1.0, 2.0, and 5.0, respectively. Table 3 presents the experimental results of four algorithms regarding the number of peaks found.

From Table 3, it can be seen that the performance of MPSO does not vary when the environmental change is not very significant, while its performance degrades a little when the environment changes significantly in terms of the shift severity. For example, the number of peaks found by MPSO is 9.72, 9.70, and 9.61, respectively, when the value of s was set to 0.5, 1.0, and 2.0, respectively, while MPSO only achieves 8.93 peaks on the average when $s = 5.0$. In addition, it is noticeable that the influence of the environmental shift severity upon MPSO is much lower than the influence upon other algorithms, which validates our expectation of our proposed speciation method for MPSO in maintaining the population diversity.

4.3 Experiments on Comparing MPSO with Peer Algorithms

The second group of experiments compares the performance of MPSO with several state-of-the-art peer algorithms in the literature, including mCPSO, mQPSO, SPSO, SOMPSO, and CPSO, on the MPB problems with different settings.

1) *Effect of varying the number of peaks*: In this set of experiments, we compare the performance of MPSO with the five peer algorithms on the MPB problems with different numbers of peaks, where the number of peaks was set to the values in the range from 5 to 50. The experimental results in terms of the offline error and standard deviation are shown in Table 4, where the results of the other five algorithms are provided by the corresponding papers with their optimal configuration that enables them to achieve their best performance.

From Table 4, it can be seen that MPSO performs much better than the peer algorithms on most

Table 5: The offline error of MPSO and the peer algorithms on the MPB problems with different shift severities

s	MPSO	SPSO	SOMPSO	CPSO	mCPSO	mQPSO
0.5	0.62±0.09	1.98±1.19	1.67±0.89	0.91±0.10	1.52±0.09	1.47±0.08
1.0	0.71±0.20	2.28±1.10	2.31±1.47	1.06±0.24	2.09±0.09	1.83±0.06
2.0	0.99±0.40	2.58±1.15	2.56±1.61	1.17±0.22	2.84±0.63	2.59±0.74
5.0	1.03±0.39	4.54±1.89	2.87±1.60	1.58±0.32	4.97±1.03	4.43±1.05

test problems. In MPSO, the proposed new speciation method can form multiple species in different search regions of the solution space and the memory-based re-initialization scheme can keep the newly generated species from re-exploring any achieved optimum, which enables MPSO to maintain the capacity of searching for new peaks. These two schemes can distribute the particles in as many as possible different promising search regions in order to ensure the algorithm locate multiple optima in dynamic environments. In addition, the adaptive LS method is used to improve the quality of the seeds of species, which can enhance the exploitation capacity of species for the optimum. Obviously, the good performance of MPSO over the other peer algorithms shows the efficiency of these designed operations in MPSO for DMMOPs.

From Table 4, it can also be seen that the performance of MPSO is affected by the number of peaks. In our experiments, the offline error of MPSO is on the average 0.56, 0.71, 1.61, and 1.58 when the number of peaks increases from 5 to 10, 20, and 50, respectively. This result is easy to understand. When the number of peaks is not very large, the species in MPSO can locate different promising search areas effectively, which has been shown in Table 2, where MPSO can almost achieve all peaks when the number of peaks is 5 and 10, and the sufficient exploitation by the LS method in MPSO can ensure the species achieve the optimum quickly. However, the situation becomes different when MPSO is applied to the MPB problem with a large number of peaks. In this case, it is very difficult for the algorithm to locate all peaks in the fitness landscape since the population size is always limited. MPSO only ensures a species to focus the search on the relatively higher peaks in its local sub-region once the species covers more than one peak in its local sub-region. It is still exciting to see that MPSO can always re-achieve a high-quality optimum in a new environment in comparison with the other four peer algorithms. Within the five peer algorithms, only CPSO can obtain the similar performance as MPSO, while SPSO, SOMPSO, mCPSO and mQPSO are beaten by MPSO significantly.

2) *Effect of varying the shift severity*: In this set of experiments, we investigate whether MPSO adapts well to different environmental dynamics in terms of the shift severity in the MPB problem. The experimental results with respect to the offline error and standard deviation are given in Table 5. The experimental results of the five peer algorithms are also taken from the corresponding papers with the algorithm settings that enable them to achieve their best performance.

From Table 5, it can be seen that the results achieved by MPSO are much better than the results of the other five algorithms on the MPB problems with different shift severities. The reason lies in that the adaptive LS method in MPSO can ensure a species to achieve one optimum efficiently once it falls within the search region of the species. This result further testifies the validity of introducing a proper LS technique into PSO for DMMOPs.

From Table 5, it can also be seen that the performance of MPSO is influenced slightly when the value of s is varied. The offline error of MPSO is on the average 0.62, 0.71, 0.99, and 1.03 when the value of change severity s changes from 0.5 to 1.0, 2.0, and 5.0, respectively. This means that MPSO can always track the new peaks quickly when an environmental change in any severity level occurs. Similar result can be observed only by SOMPSO and CPSO, while the performance of the other three peer algorithms degrades with a large degree when the environment is subject to significant changes in terms of the severity. The experimental results show that our proposed algorithm can adapt well to the environmental dynamics in terms of the shift severity.

3) *Effect of varying the environmental change period*: In this set of experiments, we examine the effect of different environmental change periods on the performance of MPSO. Table 6 presents the experimental results of all algorithms on the MPB problem with different number of peaks when the value of the change period (U) is increased from the default setting of 5000 to 10000.

Table 6: The offline error of MPSO and the peer algorithms on the MPB problems with different number of peaks and the change period 10000

peaks	MPSO	SPSO	SOMPSO	CPSO	mCPSO	mQPSO
5	0.30±0.07	0.29±0.07	0.25±0.05	0.22±0.07	0.37±0.05	0.35±0.03
10	0.50±0.06	0.52±0.10	0.64±0.12	0.63±0.06	0.53±0.08	0.50±0.04
20	1.09±0.24	1.28±0.34	2.87±0.76	1.06±0.19	1.07±0.24	0.96±0.09
50	1.18±0.26	2.12±0.63	4.36±1.13	1.05±0.14	1.74±0.26	1.61±0.11

From Table 6, it can be seen that the performance of all investigated algorithms gets much better when the value of the change period is increased. For example, the offline error of MPSO, SPSO, SOMPSO, CPSO, mCPSO, and mQPSO on the MPB problem with ten peaks is 0.50, 0.52, 0.64, 0.63, 0.53, and 0.50, respectively in Table 6, which is much lower than the corresponding value of these algorithms on the same MPB problem with the change period set to 5000, as shown in Table 4. This is because increasing the value of the change period gives algorithms more time to search before the next environmental change occurs.

From Table 6, it can also be seen that the results achieved by MPSO are similar to that achieved by other peer algorithms on most cases when the change period is set to 10000, which did not happen in the above experiments. The reason behind this is that the effect of the LS method is limited when the individual selected for local refinement has already a high fitness level.

5 Conclusions and future work

This paper investigates a PSO-based memetic computing approach (MPSO) for solving dynamic multi-modal optimization problems (DMMOPs). A new speciation method, where multiple species can be formed adaptively according to the indices of particles in the population rather than the Euclidean distance as in the speciation method in [26], is proposed to locate and track multiple optima in dynamic environments. An adaptive local search (LS) method, which employs two different LS operators in a cooperation fashion, is also designed to speed up the exploitation of one species. In order to decrease useless LS operations, an adaptive scheme that adjusts the execution probability and step size of LS automatically is applied during the LS process. In addition, a memory-based re-initialization scheme, where the best particles found by converged species can be stored in a memory pool and retrieved into the initial population in a new environment when an environmental change is detected, is applied in order to ensure MPSO adapt well to a changing multi-modal environment.

In order to justify the proposed algorithm, experiments were carried out to analyze the effect of the key mechanisms proposed in MPSO and compare the performance of MPSO with several state-of-the-art algorithms for DMMOPs (i.e., SPSO [26], SOMPSO [36], CPSO [38], mCPSO [1], and mQPSO [1]) on the widely used MPB problems.

From the experimental results, the following conclusions can be drawn. The performance of MPSO scales well regarding the number of peaks in dynamic environments. When the number of peaks in the solution space is relatively small (e.g., less than 20), MPSO performs much better than SPSO, SOMPSO, CPSO, mCPSO, and mQPSO. MPSO also adapts well to the environmental dynamics in terms of the change severity. MPSO outperforms all the other peer algorithms with a great degree on all MPB problems with different environmental shift severities. Generally speaking, MPSO greatly improves the performance of PSO in terms of tracking and locating multiple optima in a dynamic environment due to the introduction of the speciation method and the proper LS technique. The experimental results indicate that the proposed MPSO algorithm can be a good optimizer in dynamic environments, especially for DMMOPs.

There are several future works to be pursued relevant to this paper. First, the search region of one species still depends on the value of parameter r_s in our proposed speciation method, which makes it difficult to generate accurate species for the situation when a single peak is covered by only one species. Although r_s can be determined using the primary experiments in this paper, more works should be done

to solve this problem. Second, it is also valuable to carry out the sensitivity analysis on the effect of parameters (e.g. α, β, and δ) upon the proposed adaptive LS method in the future. Note that the values of these parameters are also given based on some simple preliminary experiments in this work. Finally, it would be also interesting to combine other techniques into MPSO to further improve its performance in more complex dynamic multi-modal environments.

Acknowledgement

We would like to thank the anonymous associate editor and reviewers for their thoughtful suggestions and constructive comments. The work by Hongfeng Wang and Dingwei Wang was supported by the National Nature Science Foundation of China (NSFC) under Grant 71001018, Grant 70431003, and Grant 70671020, the National Innovation Research Community Science Foundation of China under Grant 60521003, the National Support Plan of China under Grant 2006BAH02A09, the Fundamental Research Funds for the Central Universities Grant N090404020, and the Science Foundation for the Excellent Youth Scholars of Ministry of Education of China under Grant 200801451053. The work by Shengxiang Yang was supported by the Engineering and Physical Sciences Research Council (EPSRC) of UK under Grant EP/E060722/01 and Grant EP/E060722/02. The work by W. H. Ip was supported by Hong Kong Polytechnic University Research Grants under Grant G-YH60.

References

[1] T. M. Blackwell and J. Branke (2006). Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Trans. on Evolutionary Computation*, **10**(4): 459-472.

[2] J. Branke, T. Kaubler, C. Schmidt and H. Schmeck (2000). A multi-population approach to dynamic optimization problems. *Proc. of the 4th Int. Conf. on Adaptive Computing in Design and Manufacturing*, pp. 299-308.

[3] J. Branke. The moving peaks benchmark website. http://www.aifb.unikarl-sruhe.de/jbr/MovPeaks.

[4] R. Brits, A. P. Engelbrecht and F. van den Bergh (2002). Solving systems of unconstrained equations using particle swarm optimization. *Proc. of IEEE Conf. Syst., Man, Cybern.*, pp. 102-107.

[5] R. Brits, A. P. Engelbrecht and F. van den Bergh (2007). Locating multiple optima using particle swarm optimization. *Applied Mathematics and Computation*, 189: 1859-1883.

[6] H. G. Cobb (1990). An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environment. *Technical Report AIC-90-001*, Naval Research Laboratory, Washington, USA.

[7] C. A. C. Coello (2002). 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): 1245-1287.

[8] A. Caponio, F. Neri and V. Tirronen (2009). Super-fit Control Adaptation in Memetic Differential Evolution Frameworks. *Soft Computing*, 13(8): 811-831.

[9] W. Du and B. Lin (2008). Multi-strategy ensemble particle swarm optimization for dynamic optimization. *Information Science*, 178(15): 3096-3109.

[10] Y. Jin and J. Branke (2005). Evolutionary pptimization in uncertain environments-a survey. *IEEE Trans. on Evol. Comput.*, 9(3): 303-317.

[11] S. Janson and M. Middendorf (2006). A hierarchical particle swarm optimizer for noisy and dynamic environments. *Genet Program Evolvable Mach*, 7(3): 329-354.

[12] J. Kennedy (1997). The Particle Swarm: Social Adaptation of Knowledge. *Proc. of IEEE Int. Conf. on Evolutionary Computation*, pp. 303-308.

[13] J. Kennedy (2000). Stereotyping: improving particle swarm performance with cluster analysis. *Proc. of IEEE Int. Conf. on Evolutionary Computation*, pp. 1507-1512.

[14] J. Kennedy, R. C. Eberhart and Y. Shi (2001). *Swarm Intelligence*. San Francisco, CA: Morgan Kaufmann Publishers.

[15] N. Krasnogor and J. Smith (2005). A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *IEEE Trans. Evol. Comput.*, 9(5): 474-488.

[16] J. Li, M. E. Balazs, G. Parks and P. J. Clarkson (2002). A species conserving genetic algorithm for multimodal function optimization. *Evolutionary Computation*, 10(2): 207-234.

[17] D. Liu, K. C. Tan, C. K. Goh, and W. K. Ho (2007). A Multiobjective Memetic Algorthm Based on Particle Swarm Optimization. *IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics*, 37(1): 42-50.

[18] X. Liu, Y. Wu and J. Ye (2008). An Improved Estimation of Distribution Algorithm in Dynamic Environments. *Proc. of the 4th Int. Conf. on Natural Computation*, pp. 269-272.

[19] C. Li and S. Yang (2008). Fast Multi-Swarm Optimization for Dynamic Optimization Problems. *Proc. of the 4th Int. Conf. on Natural Computation*, pp. 624-628.

[20] R. Mendes and A. S. Mohais (2005). DynDE: A differential evolution for dynamic optimization problems. *Proc. of IEEE Int. Conf. on Evolutionary Computation*, pp. 2808-2815.

[21] Z. Michalewicz and M. Schoenauer (1996). Evolutionary algorithm for constrained parameter optimization problems. *Evolutionary Computation*, 4(1): 1-32.

[22] F. Neri, G. Iacca and E. Mininno (2011). Disturbed Exploitation Compact Differential Evolution for Limited Memory Optimization Problems. *Information Sciences*, 181(12): 2469-2487.

[23] F. Neri and E. Mininno (2010). Memetic Compact Differential Evolution for Cartesian Robot Control. *IEEE Computational Intelligence Magazine*, 5(2): 54-65.

[24] Y.-S. Ong and A. J. Keane (2004). Meta-lamarckian learning in memetic algorithms. *IEEE Trans. Evol. Comput.*, 8(2): 99C110.

[25] Y.-S. Ong, M. Lim, N. Zhu and K. Wong (2006). Classification of adaptive memetic algorithms: a comparative study. *IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics*, 36(1): 141-152.

[26] D. Parrott and X. Li (2006). Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Trans. on Evol. Comput.*, 10(4): 440-458.

[27] Y. G. Petalas, K. E. Parsopoulos and M. N. Vrahatis (2007). Memetic particle swarm optimization. *Ann Oper Res*, 156: 99-127.

[28] J. E. Smith (2007). Coevolving memetic algorithms: A review and progress report, *IEEE Trans. on SMC-Part B: Cybern.*, 37(1): 6-17.

[29] M. SamanPishvaee, R. Z. Farahani and W. Dullaert (2010). A memetic algorithm for bi-objective integrated forward/reverse logistics network design. *Computers & Operations Research*, 37: 1100-1112.

[30] S. Salhi and N. M. Queen (2004). A hybrid algorithm for identifying global and local minima when optimizing functions with many minima. *European Journal of Operational Research*, 155: 51-67.

[31] K. Trojanowski (2007). B-Cell Algorithm as a Parallel Approach to Optimization of Moving Peaks Benchmark Tasks. *Proc. of the 6th Int. Conf. on Computer Information Systems and Industrial Management Applications*, pp. 143-148.

[32] M. Weber, F. Neri and V. Tirronen (2009). Distributed Differential Evolution with Explorative-Exploitative Population Families. *Genetic Programming and Evolvable Machines*, 10(4): 343-371.

[33] M. Weber, V. Tirronen and F. Neri (2010). Scale Factor Inheritance Mechanism in Distributed Differential Evolution. *Soft Computing*, 14(11): 1187-1207.

[34] H. Wang, D. Wang and S. Yang (2009). A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems. *Soft Computing*, 13(8-9): 763-780.

[35] H. Wang, S. Yang, W. H. Ip and D. Wang (2009). Adaptive primal-dual genetic algorithms in dynamic environments. *IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics*, 39(6): 1348-1361.

[36] H. Wang, S. Yang, W. H. Ip and D. Wang (2010). A particle swarm optimization based memetic algorithm for dynamic optimization problems. *Natural Computing*, 9(3): 703-725.

[37] L. Xing, Y. Chen and H. Cai (2006). An intelligent genetic algorithm designed for global optimization of multi-minima functions. *Applied Mathematics and Computation*, 178: 355-371.

[38] S. Yang and C. Li (2010). A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments. *IEEE Trans. on Evol. Comput.*, 14(6): 959-974.

[39] S. Yang, Y.-S. Ong, and Y. Jin (eds.), *Evolutionary Computation in Dynamic and Uncertain Environments*, Springer-Verlag, Berlin Heidelberg, 2007.

[40] S. Yang and X. Yao (2008). Population-based incremental learning with associative memory for dynamic environments. *IEEE Trans. on Evol. Comput.*, 12(5): 542-561.

[41] J. Zhang, D. Huanga, T. M. Lok and M. R. Lyu (2006). A novel adaptive sequential niche technique for multimodal function optimization. *Neurocomputing*, 69: 2396-2401.

[42] E. Zitzler and L. Thiele (2006). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans. Evol. Comput.*, 3(4): 257-271.