Natural Computing
Manuscript number to be inserted by the editor.

A Particle Swarm Optimization Based Memetic Algorithm for Dynamic Optimization Problems

Hongfeng Wang[1,4], Shengxiang Yang[2], W. H. Ip[3], Dingwei Wang[1]

[1] School of Information Science and Engineering, Northeastern University
Shenyang, P. R. China
e-mail: {hfwang, dwwang}@mail.neu.edu.cn
[2] Department of Computer Science, University of Leicester
University Road, Leicester LE1 7RH, United Kingdom
e-mail: s.yang@mcs.le.ac.uk
[3] Department of Industrial and Systems Engineering, Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong, P. R. China
e-mail: mfwhip@inet.polyu.edu.hk
[4] Department of Industrial Engineering, Pusan National University, Pusan, Korea.

November 9, 2011

Abstract Recently, there has been an increasing concern from the evolutionary computation community on dynamic optimization problems since many real-world optimization problems are dynamic. This paper investigates a particle swarm optimization (PSO) based memetic algorithm that hybridizes PSO with a local search technique for dynamic environments. Within the framework of the proposed algorithm, a local version of PSO with a ring-shape topology structure is used as the global search operator and a fuzzy cognition local search method is proposed as the local search technique. In addition, a self-organized random immigrants scheme is extended into our proposed algorithm in order to further enhance its exploration capacity for new peaks in the search space. Experimental study over the moving peaks benchmark problem shows that the proposed PSO-based memetic algorithm is robust and adaptable in dynamic environments.

1 Introduction

In recent years, dynamic optimization problems (DOPs) have attracted a growing interest from the evolutionary computation community since many real-world problems are time-varying. In dynamic environments, the optimization goal, problem instance, and/or some constraints may change over time due to such factors as stochastic arrival of new jobs, machine faults and degradation, and so on. In such cases, the goal of an optimization algorithm is no longer to find a satisfactory solution to a fixed problem, but to track the moving optimum in the search space

as closely as possible. This poses great challenges to traditional evolutionary algorithms (EAs) because they cannot adapt well to the changing environment once converged. Over the past decade, a number of approaches [3,5,7,30,32–34,36,37, 39,40] have been developed into EAs for DOPs and readers are referred to [15,38] for a comprehensive overview.

Memetic algorithms (MAs) are a recent growing area of research in evolutionary computation. MAs combine EAs with local search (LS) methods and hence are also referred to as genetic local search [18,26]. In the framework of MAs, EA operators are used for globally rough search and LS operators are used for local improvement, which can maintain an efficient balance between exploration and exploitation of an algorithm. MAs have been applied widely on many complex optimization problems, such as scheduling problems [14,21,23], combinatorial optimization problems [9,27, 28], multi-objective problems [11,13,20], and so on. However, it is noticeable that the problems for which MAs have been applied are mainly stationary problems. MAs are rarely applied in dynamic environments.

Particle swarm optimization (PSO), which simulates the social behaviour of a group of fishes or birds for solving problems, has become another rapidly growing active topic over the last decade. PSO has been applied for many optimization problems with promising results. In the recent years, PSO has been applied to address dynamic environments [2,24]. Based on the different learning approaches of particles, PSO comes with two versions, the global version and the local version. In the global PSO, each particle learns from the best particle in the whole swarm while in the local version each particle learns from the best particle in its neighborhood. Of these two versions, the local PSO has a slower convergence speed and hence may adapt to a changing environment more easily.

In this paper, a new PSO-based MA, which hybridizes a local version of PSO and a *fuzzy cognition local search* method, is proposed to address DOPs. In addition, a *self-organized random immigrants* scheme is also integrated into the proposed algorithm in order to further improve its performance in dynamic environments. Based on the moving peaks benchmark (MPB) problem [**?**], a set of extensive experiments are carried out to examine the major features of the proposed PSO-based MA and to compare its performance with some peer algorithms from the literature in dynamic environments.

The rest of this paper is outlined as follows. The next section reviews the basic principles of PSOs and existing related work on PSOs for DOPs. Section 3 describes the proposed PSO-based MA in details. Section 4 evaluates empirically the proposed algorithm on the moving peaks benchmark in comparison with several alternative approaches from the literature and some experimental analyses on parameter settings are also presented. The final section concludes this paper with discussions on future work.

2 Related Work

2.1 Principles of Particle Swarm Optimization

Similar to EAs, PSO is a population-based, iterative technique. The main difference lies in that in PSO potential solutions (particles) move, rather than evolve, through

the search space. The rules, which govern this movement, are inspired by the social interaction among a school of fishes or flock of birds in nature. In a PSO model, a particle can be represented by its position and velocity. At every iteration, each particle in a population (swarm) 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:

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

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

where $\mathbf{v}_i(t)$ and $\mathbf{x}_i(t)$ represent the current velocity and position of particle i at iteration t, respectively, $\mathbf{p}_i(t)$ is the position of the best solution (*pbest*) found so far by particle i, $\mathbf{p}_{gi}(t)$ 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 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, global and local. In the global version of PSO algorithms, each particle can be influenced by the best fitness particle in the whole population which causes that all particles move and converge quickly into one optimum point in the search space. On the contrast, the local version of PSO only allows each particle to be influenced by the best fitness particle chosen from its neighborhood which makes the algorithm exhibit a good exploration capacity due to its slow convergence. Considering that a slowly converging population can adapt to a changing environment more easily, it becomes an interesting research issue to examine the performance of the local variant of PSO, which is hybrized with suitable LS techniques, in dynamic environments.

2.2 PSO in Dynamic Environments

The work on PSO for addressing DOPs was first reported by Eberhart and Shi in [8], where a conventional PSO was investigated to track a single peak that varies spatially. Their experimental results show that the simple PSO can not deal with a variety of changes. Hu and Eberhart [12] introduced an adaptive PSO, which automatically tracks various changes in a dynamic system. They tested different environmental detection and re-randomization strategies, which effectively respond to a wide variety of changes, and reported and analyzed the experimental results on the parabolic function and Rosenbrock's benchmark function with various severities of environmental changes. A method of adapting PSO for dynamic environments was present by Carlisle and Dozier [6]. In their PSO, each particle can reset the record of its best position and avoid making the direction and velocity change decisions based on outdated information when the environment has changed. Two resetting methods were examined and the experimental results show that both were able to improve the performance of PSO in both static and dynamic environments.

Recently, some special diversity schemes have been developed for PSO in dynamic environments. Blackwell and Bentley [1] introduced a charged PSO, where a nucleus of neutral particles is surrounded by some charged particles. The charge imposes a repulsion force between particles and thus hinders the swarm to converge.

In [22], a composite PSO was proposed to address dynamic environments, where composite particles are constructed as a novel type of particles in the search space and their motions are integrated into the swarm. A special reflection scheme is also introduced the compound PSO in order to explore the search space more comprehensively and hence enhances the performance of PSO in dynamic environments.

The multi-swarm scheme is also a common strategy for PSO in dynamic environments. Blackwell and Branke [2] used 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. On the MPB problem with ten optima, multi-quantum swarm outperforms charged particle and standard PSO. Another multi-swarm approach was recently proposed by Parott and Li [24], where the number and size of swarms are adjusted dynamically by a speciation mechanism, which was originally proposed for finding multiple optima in multi-modal landscapes. The experimental results indicate that the species-based PSO is very effective in dealing with multi-modal optimization functions in both stationary and dynamic environments. Janson and Middendorf [16] proposed a partitioned hierarchical PSO, which can maintain a hierarchy of particles that are partitioned into several sub-swarms for a limited number of generations after a change of the environment occurs.

2.3 PSO Based MAs

The algorithms investigated in this paper are a class of PSO-based MAs, which hybridize PSO algorithms with LS techniques. The general PSO-based MA for a maximization problem can be expressed by the pseudo-code in Fig. 1, where s_size denotes the population size. Within this MA, a population of s_size particles are first initialized with random positions and velocities and evaluated via a fitness function $f(\cdot)$. The *pbest* of each particle is set equal to itself at the initial step. Then, some particles, selected from the current population using a certain choice strategy, are improved by a meme or LS method, and finally the *gbest* of each particle is updated. At each subsequent iteration, particles can accomplish their update of velocities and positions according to Eqs. (1) and (2) respectively. If one particle can achieve a better solution than its *pbest*, its *pbest* will be replaced by the newly achieved solution. Finally, some particles are selected to execute local improvement before the *gbest* of each particle is updated.

Obviously, how to select suitable particles from the current population for local improvement and how to execute an effective LS for the selected particles must be addressed when we design a PSO-based MA. Moreover, the convergence problem should also be considered when it is applied for DOPs. Many diversity keeping approaches have proved to be good choices for EAs to address this problem. However, how to balance between LS and diversity keeping schemes under the framework of a PSO-based MA in dynamic environments becomes a very interesting work since both the two schemes cost a number of extra evaluations.

```
Procedure General PSO-based MA:
begin
  for i := 0 to s_size − 1 do
    initialize particle i with random position x_i and velocity v_i;
    evaluate f(x_i);
    p_i := x_i;
  endfor
  if LS is applied then
    select particles from the current population for local refinement;
    apply LS upon each of selected particles;
  endif
  for i := 0 to s_size − 1 do
    update p_gi;
  endfor
  repeat
    for i := 0 to s_size − 1 do
      update v_i and x_i according to Eqs.(1) and (2) respectively;
      evaluate f(x_i);
      if f(p_i) < f(x_i) then p_i := x_i; endif
    endfor
    if LS is applied then
      select particles from the current population for local refinement;
      apply LS upon each of selected particles;
    endif
    for i := 0 to s_size − 1 do
      update p_gi;
    endfor
  until a stop condition is met
end
```

Fig. 1 Pseudo-code for a general PSO-based MA.

In the next section, some approaches will be proposed and discussed in detail to address the aforementioned problems, which result in the proposed PSO-based MAs for DOPs.

3 Proposed PSO Based MAs

3.1 The PSO Topology Structure

As discussed in the above section, a local version of PSO can reduce, at least temporarily, the pressure of convergence of the population and thus enable the algorithm maintain a sufficient exploration capacity for a longer time. This is necessary for adapting the PSO well to a changing environment. Here, the first question to ask for this kind of PSO algorithms is which topology structure should be used, i.e., how to choose the neighbors of a particle. The approaches to design the local neighborhood of a particle can be grouped into two categories as follows.

The local neighborhood of the first category is defined according to the actual distance between the particles in the search space, which is translated into the Euclidean distance for genotype with a real-coded representation, or the Hamming distance for genotype using a binary representation. Parrot and Li [24] proposed the notion of a species, which was designed to be a group of particles that fall within a certain distance from a species seed, i.e., the particles with the best fitness in a species. The whole population can be divided into some different species and particles within the same species choose the species seed as their neighborhood best *gbest*. The disadvantage of approaches in this category is that when choosing a particle's neighbors, the distance between the particle and other particles need always to be re-calculated.

In the second category, the neighboring particles are identified using their indices rather than their genotype distance. The particles are often lied on a ring-shaped topology, i.e., the particle $\mathbf{x}_0$ is the immediate neighbor of the particle $\mathbf{x}_{s\_size-1}$. If r_n is the neighborhood's radius, particle i is neighbored by the particles with indices from $((i - r_n)\%s\_size)$ to $((i + r_n)\%s\_size)$. Similarly, Janson and Middendorf [16] introduced a hierarchical PSO algorithm where each particle is arranged in a node of a rooted tree and only neighbored to the particle that is in its parent node. A local variant of PSO with a grid-like neighborhood structure is also tested by Li and Dam [19].

Here, the ring topology structure of PSO is adopted in our investigated algorithm and the radius of neighborhood r_n is set to 1, which means each particle only communicates with two neighbors whose indices are closest to it. It is obvious that the smaller the value of r_n is, the more slowly the population converges. We choose $r_n = 1$ because it is always helpful to maintain the population diversity with a great degree in dynamic environments.

3.2 Local Search

Within MAs, LS operators are employed to improve the individuals' quality efficiently in a local area. In the context of PSO-based MAs, a LS improvement procedure can often be regarded as the particle executing one iterative biased or random moving, while the moving from the current solution to a candidate solution would be accepted if the candidate has a better fitness at each iterative step. The general LS operator in PSO-based MAs can be illustrated by the pseudo-code in Fig. 2.

From Fig. 2, it can be seen that the primary factor that affects the behavior of the LS operator is the choice of the velocity ($\mathbf{v}'$) generating method. Petalas et al. [25] employed a stochastic iterative LS technique in their MA, called *random walk with direction exploitation* (RWDE), where a sequence of approximations of the optimizer are generated by assuming a random vector as a search velocity. Liu et al. [20] presented a *synchronous particle local search* (SPLS) method where a velocity vector is generated through the conventional particle velocity formula except for that *gbest* is replaced with a fuzzy vector. Inspired by these works, a *fuzzy cognition local search* (FCLS), which utilizes a special *cognition-only model* of PSO to generate a velocity vector, is proposed in this paper.

Kennedy [17] introduced two special models of PSO, which are defined by omitting or restricting components in the velocity formula. Dropping the social component results in the cognition-only model, whereas dropping the cognitive component

```
Procedure General LS Operator:
begin
  x' := x;
  for j := 1 to ls_size do
    generate a velocity vector v';
    x' := x + v';
    if f(x') > f(x) then x := x'; endif
    if f(x') > f(p) then p := x'; endif
  endfor
end;
Denotations:
  x: the selected particle for local improvement
  ls_size: the step size of the LS operator
  v': the velocity of each step the LS moves
  p: pbest of a selected particle x
```

Fig. 2 Pseudo-code for the general LS operator in PSO-based MAs.

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 FCLS operator based on the cognition-only PSO model.

In order to improve the quality of a particle as much as possible, a *fuzzy cognition-only model* is proposed here, which is formulated as follows.

$$\mathbf{p}'(t) = N(\mathbf{p}(t), \sigma) \tag{3}$$

$$\mathbf{v}(t+1) = \omega\mathbf{v}(t) + c_1\xi(\mathbf{p}'(t) - \mathbf{x}(t)), \tag{4}$$

where $\mathbf{p}'(t)$ denotes a fuzzy position of *pbest* ($\mathbf{p}(t)$) of particle $\mathbf{x}$ at time t, which is characterized by a normal distribution vector $N(\mathbf{p}(t), \sigma)$ with σ being a predefined parameter to control the distribution range.

Based on the fuzzy cognition-only model, the FCLS procedure can be expressed by the pseudo-code shown in Fig. 3. From the pseudo-code, it can be seen that σ is an important parameter that affects the FCLS operator because the move of a particle is directed by the fuzzy position of *pbest*. The smaller the value of σ is, the closer the position of the generated fuzzy *pbest* approaches its original position. This means that the particle can execute one biased move towards a smaller area around *pbest*. When $\sigma = 0$, the particle's move is decided by its *pbest* absolutely as a result that the *fuzzy cognition-only model* returns to the cognition-only model. Moreover, the initialization of $\mathbf{v}'$ (see Fig. 3) is also confined within a small range ($\mathbf{v}'$ is initialized within the range of $[0, 5]^n$ in this paper) in order to reduce the influence of the random factor.

The next problem to be addressed is how to select suitable particles from the current population for local improvement (see Fig. 1). In many researches, LS is performed on each of newly generated individuals, which seems to be too costly and infeasible for a MA for DOPs considering that the total cost per iteration in terms of function evaluations is limited. Hence, it seems a good choice that only

```
Procedure Algorithm of FCLS operator:
begin
  x' := x;
  initialize v' within a small range;
  for j := 1 to ls_size do
    generate p' according to Eq. (3);
    generate v' according to Eq. (4);
    x' := x + v';
    if f(x') > f(x) then x := x'; endif
    if f(x') > f(p) then p := x'; endif
  endfor
end;
The denotations are the same as in Fig. 2
```

Fig. 3 Pseudo-code for the FCLS operator.

```
Procedure Algorithm of selecting particles for LS:
begin
  S := ∅;
  count := 0;
  P' := P;
  while (count <= max_ls_count) do
    get the best fitness particle x ∈ P';
    S := S ∪ {x};
    count := count + 1;
    remove x from P';
    for each y ∈ P' do
      if d(x, y) < r_s then remove y from P'; endif
    endfor
    if P' = ∅ then break; endif
  endwhile
end;
Denotations:
  S: a list of all selected particles to be improved by LS
  P: the current population
  max_ls_count: the maximal allowable value of |S| per iteration
  d(x, y): the Euclidean distance between two particles x and y
  r_s: a constant that controls the crowded degree in S
```

Fig. 4 Pseudo-code for the choice algorithm that selects particles for LS.

the best fitness individual being selected for local improvement [35], if there exists a single optimum in the search space. However, if the dynamic multi-modal function is considered, this scheme may become unsuitable. Here, a selection algorithm, which can ensure multiple high-quality, non-crowded particles to be refined by LS simultaneously, is illustrated in Fig. 4.

The algorithm (as shown in Fig. 4) of selecting particles for LS is performed at each iteration. The set S of particles for LS is initially set to null and all particles in the current population P are copied into a temporary choice pool P'. The best fitness particle $\mathbf{x}$ is firstly selected from P' into S and then the remainders in P' are checked in turn against $\mathbf{x}$. If a particle falls within the radius r_s of $\mathbf{x}$, it is removed away from P'. This choice course will be iterated until P' becomes a null set or the maximal allowable number of LS operations is met. It is easy to understand that this algorithm aims to pick out multiple high-quality particles in different regions of the search space. The parameter r_s can control the crowded degree of S and ensure LS exploit different peaks in the search space. In addition, at each iteration, the LS step size ls_size for each particle in S needs to be re-calculated because the size of S, i.e., $|S|$, may be different at different iteration steps. In the experiments reported later in this paper, $ls\_size = \lceil total\_ls\_size/|S| \rceil$, where $total\_ls\_size$ denotes the total LS step size for S per iteration.

3.3 Increasing Population Diversity

Although a local version of PSO can maintain a certain degree of diversity, it can loose the capacity of exploring new regions of the search space along with its slow convergence. This situation must be avoided since the fitness landscape may change over time and new peaks may also appear in the search space over time in dynamic environments.

Some diversity-increasing approaches have been developed for EAs to address this problem. For example, the random immigrants approach is a very simple and natural way to increase the diversity level of the population. In a random immigrants genetic algorithm (RIGA) [10], a fraction of the current population is replaced by randomly generated individuals in each generation of the run. A replacement strategy, like replacing random or worst individuals of the population, defines which individuals are replaced by the immigrants. However, this simple scheme can not often improve the performance of algorithms in dynamic environments because the newly generated immigrants hardly survive during the selection process into the next generation due to their very low fitness.

Recently, a variant of RIGA, called *self-organized random immigrants genetic algorithm* (SORIGA), was proposed in [29]. In SORIGA, the worst fitness individual and some of its neighbors are replaced by random individuals and preserved into a sub-population. This sub-population is allowed to evolve independently. That is, it is subjected to selection, crossover, and mutation among individuals that belong to the sub-population. The individuals in the current population that do not belong to the sub-population are not allowed to compete with (and hence replace) the individuals of the sub-population. The sub-population may expand or shrink dynamically and may become extincted when the worst fitness individual does not fall within it.

Here the mechanism of self-organized random immigrants (SORI) in [29] is extended into the proposed PSO-based MA for DOPs. In this extension, a set of separate particles (the size is denoted as r_r), which form a sub-population, are selected randomly to be re-initialized and partitioned from the ring structure of the population every fixed number (t_ri) of iterations. The partitioned sub-population always keeps searching for the optimum independently until when a new sub-population

```
Procedure Proposed PSO-Based MA:
begin
  for i := 0 to s_size − 1 do
    initialize particle i with random x_i and v_i;
    evaluate f(x_i);
    p_i := x_i;
  endfor
  if the FCLS operator is used then
    select a set |S| of particles for local refinements;
    apply the FCLS operator upon each particle in S;
  endif
  for i := 0 to s_size − 1 do
    update p_gi;
  endfor
  repeat
    for i := 0 to s_size − 1 do
      update v_i and x_i according to Eqs.(1) and (2) respectively;
      evaluate f(x_i);
      if f(p_i) < f(x_i) then p_i := x_i;
    endfor
    if the SORI scheme is used and t%t_ri == 0 then
      set the status of each particle in the population as non-preserved;
      k := rand(0, s_size − 1);
      re-initialize the particles with indices from k to k + r_r − 1
        randomly and set their status as preserved;
    endif
    if the FCLS operator is applied then
      apply the FCLS operator upon the best fitness preserved particle;
      select a set |S| of non-preserved particles for local refinements;
      apply the FCLS operator upon each particle in S;
    for i := 0 to s_size − 1 do
      update p_gi;
    endfor
  until a termination condition is met
end
```

Fig. 5 Pseudo-code for the proposed PSO-based MA.

is generated, it is rejoined into the main population. Moreover, the sub-population accomplishes its update in a global version and its best fitness particle can be improved by a LS operator in each iteration of running the algorithm in order to ensure that it converges into a promising area in the search space quickly. Re-initialization of a particle also involves resetting its *pbest* besides randomly initializing its position and velocity.

Based on the above description, our proposed MA that hybridizes both the FCLS operator and the SORI scheme with a local version of PSO in a ring-shape topology can be summarized by the pseudo-code shown in Fig. 5. Within the proposed algo-

rithm, after an initial population is generated randomly, a set of particles, selected from the population by the choice algorithm in Fig. 4, are improved by the FCLS operator and then the *gbest* of each particle would be updated by the best particle among its neighbors. At each subsequent iterative step, each particle is firstly updated according to the update formula (Eqs. (1) and (2)), and then a portion of separate particles, selected randomly and partitioned from the population, are re-initialized and set as preserved to search for the new optima independently if the condition of SORI scheme is reached ($t\%t\_ri == 0$). Finally, the best preserved particle and a set of non-preserved particles are selected to execute local improvement before the *gbest* of each particle is updated.

4 Experimental Study

4.1 Algorithm Test Environments

In this paper, a series of dynamic test environments are generated based on the MPB problem [4] that consists of a number of peaks, of changing heights, widths, and locations in a random direction. The base landscape of the MPB function consists of m peaks defined in an n-dimensional real space as follows:

$$F(\mathbf{x}, t) = \max_{i=1,\ldots,m} \frac{H_i(t)}{1 + W_i(t) \sum_{j=1}^{n} (x_j(t) - X_{ij}(t))^2}, \tag{5}$$

where $W_i(t)$ and $H_i(t)$ 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.

The parameter settings of the MPB problem used in this paper correspond to Scenario 1 as specified on the benchmark website [4]. The test function has 5 peaks defined on a 5-dimensional real space. Every τ iterations, 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 $\mathbf{v_i}$ of a fixed length ρ. More formally, a change of a single peak can be described as follows:

$$\begin{cases} \delta \in N(0,1) \\ H_i(t) = H_i(t-1) + 7 \cdot \delta \\ W_i(t) = W_i(t-1) + 0.01 \cdot \delta \\ \mathbf{X}_i(t) = \mathbf{X}_i(t-1) + \mathbf{v}_i(t), \end{cases} \tag{6}$$

$$\mathbf{v}_i(t) = \frac{\rho}{|\mathbf{r} + \mathbf{v}_i(t-1)|}((1-\lambda)\mathbf{r} + \lambda \mathbf{v}_i(t-1)), \tag{7}$$

where the shift vector $\mathbf{v}_i(t)$ is a linear combination of a random vector $\mathbf{r}$ and the previous shift vector $\mathbf{v}_i(t-1)$ and is normalized to length ρ. The random vector $\mathbf{r}$ is created by drawing random numbers for each dimension and normalizing its length to ρ. Hence, the parameter τ may be used to control the speed of changes and ρ may be used to control the severity of changes. The parameter λ allows controlling whether changes exhibit a trend (λ is always set to 0 in our experiments).

Table 1 The index table for dynamic parameter settings

τ	Environmental Dynamics Index			
50	1	2	3	4
100	5	6	7	8
200	9	10	11	12
$\rho \rightarrow$	0.1	1.0	2.0	5.0

To test the performance of algorithms in different dynamic environments, a series of DOPs are constructed using the MPB problem. The change severity parameter ρ is set to 0.1, 1.0, 2.0, and 5.0 respectively in order to examine the performance of algorithms in dynamic environments with different severities: from slight change ($\rho = 0.1$) to moderate variation ($\rho = 1.0$ or 2.0) to intense change ($\rho = 5.0$). The change speed parameter τ is set to 50, 100, and 200 respectively, which means that the environment changes fast, in the moderate speed, and very slowly, respectively. In total, a series of 12 different DOPs are constructed and summarized in Table 1.

4.2 Experimental Design

In this section, experiments are carried out in order to study the major features of our proposed PSO-based MAs and to compare their performance with several existing peer algorithms. The following abbreviations represent the algorithms considered in this paper.

- LPSO: a simple local version of PSO, which utilizes a ring topology with the neighbor radius set to 1 (see Section 3.1);
- GPSO: a standard global version of PSO;
- LPSO-FCMA: LPSO-based MA with the FCLS operator;
- LPSO-RWMA: LPSO-based MA with the RWDE operator [25];
- GPSO-FCMA: GPSO-based MA with the FCLS operator;
- multiCPSO: a multi-swarm PSO proposed by Blackwell and Branke in [2]. In multiCPSO, the population is split into a set of sub-swarms, which interact with each other locally by an exclusion parameter and globally through a new anti-convergence operator. In addition, each sub-swarm has some charged particles, whose update is affected by the inter-particle repulsions.
- multiMPSO: a multi-swarm PSO that is similar to multiCPSO except that each sub-swarm is now equipped with the FCLS operator, rather than charged particles. In each iteration of running multiMPSO, it is firstly checked whether all sub-swarms have converged using an anti-convergence operator and if so, the worst sub-swarm will be re-initialized. Then, all pairs of sub-swarms are also tested to see whether they are too close using an exclusion parameter and if so, the inferior sub-swarm is re-initialized. Finally, all particles are simply updated and the best fitness particle in each sub-swarm is improved by the FCLS operator.

The total number of evaluations per iteration is always fixed to 100 for all algorithms in order to make an impartial comparison among them. For all PSO models, the cognitive and social learning factors c_1 and c_2 are both set to 1.4962

and the inertia weight ω is set to 0.72984 as suggested by den Bergh [31]. 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 special parameters in our proposed MAs are set as follows: $r_s = 1.0$, $r_r = 5$, and $t\_ri = 25$. The setting of other relevant parameters will be given and discussed in the experiments later on.

For each experiment of an algorithm on a test problem, 30 independent runs were executed with the same set of random seeds. For each run of an algorithm on a DOP, 10 environmental changes were allowed and the best fitness of any particle in the population is recorded. Given that the optimum fitness is always known for the test problem used in this paper, the overall offline performance of an algorithm is defined as the mean error averaged across the number of total runs and then averaged over the data gathering period, as formulated below:

$$\overline{E}_{BG} = \frac{1}{G}\sum_{i=1}^{G}(\frac{1}{N}\sum_{j=1}^{N}(F_{ij}^{*} - F_{BG_{ij}})), \tag{8}$$

where G is the number of generations (i.e., $G = 10 * \tau$), $N = 30$ is the total number of runs, F_{ij}^{*} and $F_{BG_{ij}}$ are the fitness of global optimum and the best particle in the population of iteration i of run j respectively.

4.3 Experimental Study on the Effect of LS Operators

In the experimental study on the effect of LS operators, we first study the influence of different settings of σ in FCLS in order to determine a robust setting for this parameter. In particular, we have implemented three LPSO-FCMAs, where σ is set to 0.05, 0.5, and 5.0 respectively, just on the stationary period of the MPB problem. The population size s_size is set to 90 for all algorithms and the FCLS operator within them is always applied $ls\_size = 10$ steps upon the best particle in the current population at each iteration. For each run of an algorithm, the maximum allowable number of iterations was set to 100. The experimental results are shown in Fig. 6, where the data were averaged over 30 runs.

From Fig. 6, it can be seen that LPSO-FCMA with $\sigma = 5.0$ performs the best at the early searching stage, while is beaten by LPSO-FCMA with $\sigma = 0.5$ quickly. When the value of σ is very small ($\sigma = 0.05$), LPSO-FCMA does not perform well until the late iteration stage of the algorithm running. Hence, σ is always set to 0.5 in the following experiments since a moderate value of σ always shows an adaptive capacity at the different search stages on this test problem.

Similar experiments are carried out to investigate the performance of PSO-based MAs with different LS operators, with the aim of examining the effect of the FCLS operator proposed in Sec. 2. In particular, five different algorithms, including GPSO-FCMA, LPSO-FCMA, LPSO-RWMA, LPSO, and GPSO, are tested also on the stationary period of the MPB problem. The population size s_size is set to 100 for LPSO and GPSO, but 90 for all MAs because the LS operator within them is always applied $ls\_size = 10$ steps upon the best particle in the current population at each iteration. The experimental results are shown in Fig. 7, where the data were also averaged over 30 runs.

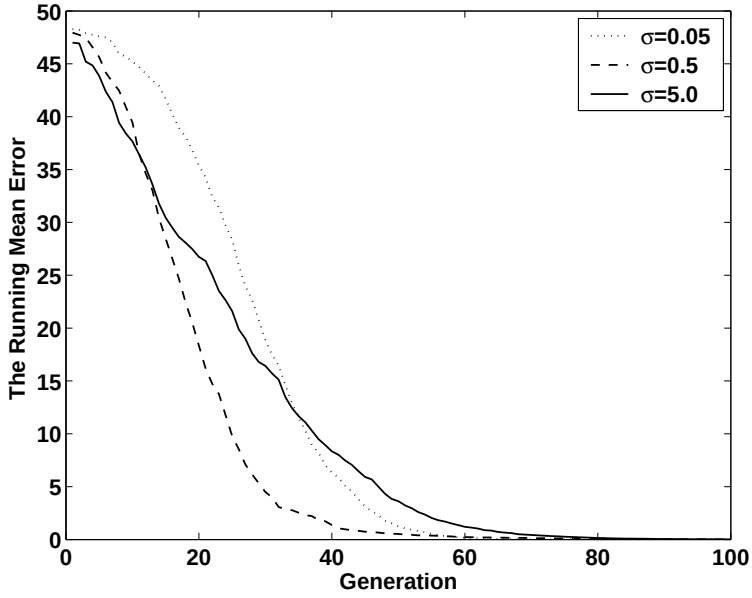

Fig. 6 Experimental results of LPSO-FCMA with different settings of σ on the stationary MPB problem.

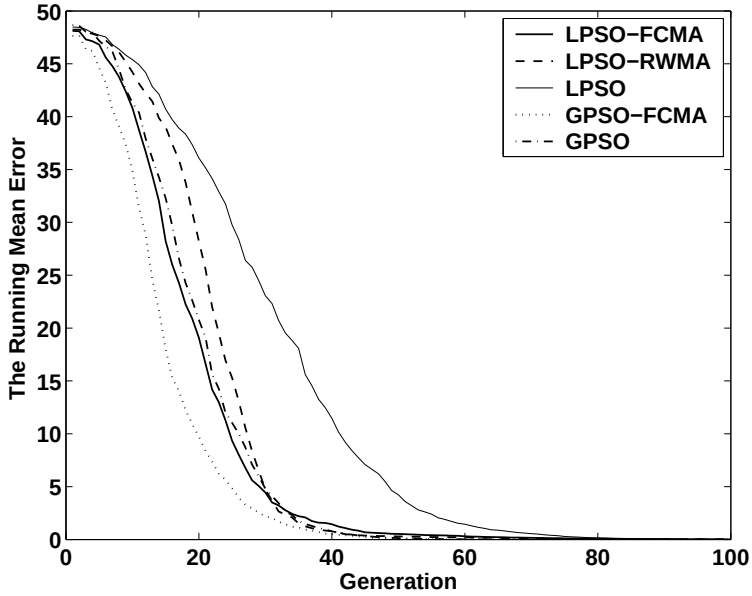

Fig. 7 Experimental results of LPSO-based MAs with different LS operators on the stationary test problem

From Fig. 7, the following results can be obtained. First, the LS operator does improve the performance of an algorithm on the stationary problem significantly. This can be seen from the results that both LPSO-FCMA and LPSO-RWMA outperform LPSO and GPSO-FCMA outperforms GPSO with a high degree. In addition, it is also shown that hybridizing LPSO with LS is very necessary from the experimental results that GPSO performs much better than LPSO, which means that the exploitation capacity of LPSO is very weak, that is, LPSO cannot achieve the optima with a higher accuracy quickly, though it can exhibit the good exploitation capacity due to its slow convergence. Second, within the two LPSO-based MAs, the FCLS operator can help them make more efficient local refinements than the RWDE operator does. The RWDE operator always executes random moves, while the FCLS operator combines the features of both random moves and biased moves, which can make the particle approach the optimum more easily. Obviously, the result that LPSO-FCMA outperforms LPSO-RWMA in this experiment validates our expectation of the proposed FCLS operator.

Another feature of the FCLS operator is how to select suitable particles from the current population for local improvement. In Sec. 3.2, we introduced a choice algorithm (see Fig. 4) that aims to select multiple high-quality, non-crowded particles simultaneously. In the following experiment, we test the effect of this choice algorithm through comparing with other two strategies: one is to improve the best fitness particle only and the other is to select a certain number of particles randomly for local refinement. For the sake of conveniently describing this experiment, LPSO-FCMA1, LPSO-FCMA2, and LPSO-FCMA3 are used to denote LPSO-FCMA with the choice algorithm in Sec. 3.2, LPSO-FCMA with choosing the best fitness particle scheme, and LPSO-FCMA with the random choice scheme respectively.

We run the above three LPSO-FCMAs on the DOPs constructed in Table 1. The population size s_size is set to 90 for LPSO-FCMA2 and 70 for LPSO-FCMA1 and LPSO-FCMA3 considering that $ls\_size = 10$ in FCMA2, the number of selected particles is set to 5 and ls_size is set to 6 in LPSO-FCMA3 and the maximum allowable number (max_ls_count) of particles selected for LS is set to 5 and the total step size of LS per iteration is set to 30 in LPSO-FCMA1. The experimental results with respect to the overall offline performance are presented in Table 2 and plotted in Fig. 8. The corresponding statistical results of comparing algorithms by the one-tailed t-test with 58 degrees of freedom at a 0.05 level of significance are given in Table 3. In Table 3, the t-test result regarding Alg. 1 - Alg. 2 is shown as "$s+$", "$s-$", "$+$", or "$-$" when Alg. 1 is significantly better than, significantly worse than, insignificantly better than, or insignificantly worse than Alg. 2 respectively.

From Table 2, Table 3 and Fig. 8, several results can be observed. In general, LPSO-FCMA1 always performs the best on most dynamic problems, which shows the validity of our proposed choice algorithm in improving the performance of LPSO-based MAs in dynamic environments. The reason is that LPSO-FCMA1 can exploit multiple particles that are distributed in different peaks of the fitness landscape synchronously using the FCLS operator, which is very helpful in a changing multimodal environment.

LPSO-FCMA2 performs better than LPSO-FCMA3, even outperforms LPSO-FCMA1 on several dynamic problems when the change severity parameter ρ is small. This is because a new optimum can be close to the previous one when ρ

Table 2 Experimental results with respect to overall offline performance of LPSO-FCMAs on dynamic test problems

Dynamics		Algorithms		
τ	ρ	LPSO-FCMA1	LPSO-FCMA2	LPSO-FCMA3
50	0.1	1.97±0.90	2.30±1.83	3.24±1.47
50	1.0	2.63±1.06	2.85±1.14	3.59±1.32
50	2.0	3.42±1.04	3.94±1.88	5.17±1.49
50	5.0	5.52±1.04	6.27±1.42	9.26±1.87
100	0.1	1.04±0.91	1.00±0.26	1.56±0.66
100	1.0	1.38±0.68	1.27±0.58	2.01±0.95
100	2.0	2.13±1.47	2.46±1.61	2.79±1.03
100	5.0	3.37±0.50	4.01±1.24	5.24±0.78
200	0.1	0.46±0.25	0.53±0.51	0.87±0.67
200	1.0	0.77±0.61	0.82±0.62	1.03±0.51
200	2.0	1.12±0.62	1.36±0.76	1.51±0.51
200	5.0	1.88±0.43	2.78±1.15	3.04±0.87

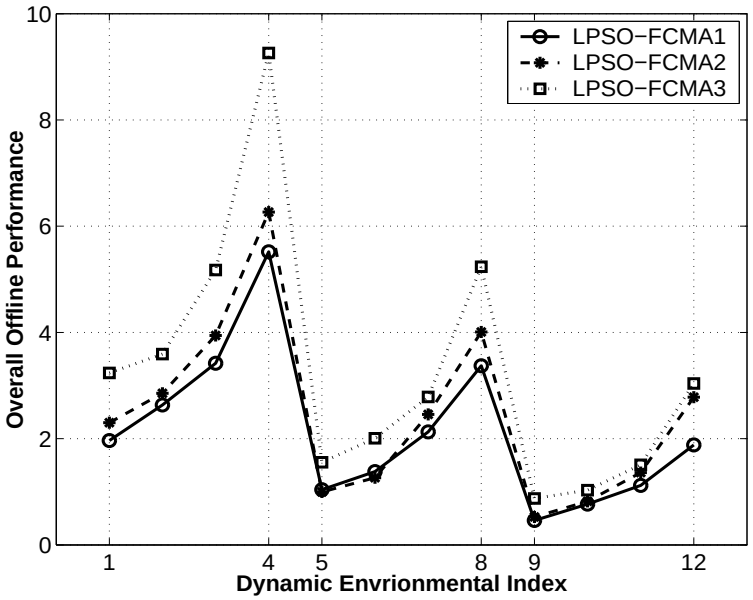

Fig. 8 Experimental results with respect to the overall offline performance of LPSO-FCMAs on the dynamic test problems.

is very small. For such instances, executing sufficient FCLS operations only for the best fitness particle may be more beneficial. However, once the environment is subject to significant changes, the choice algorithm in LPSO-FCMA2 can mislead the FCLS operator. This is why LPSO-FCMA2 performs significantly worse than LPSO-FCMA1 on all dynamic problems when $\rho = 5.0$, see the relevant t-test results regarding LPSO-FCMA1−LPSO-FCMA2 in Table 3. LPSO-FCMA3 always

Table 3 The t-test results of comparing LPSO-FCMAs regarding the overall offline performance on the dynamic test problems

t-test Result	MPB Problem			
$\tau = 50$, $\rho \Rightarrow$	0.1	1.0	2.0	5.0
LPSO-FCMA1 − LPSO-FCMA2	+	+	+	$s+$
LPSO-FCMA1 − LPSO-FCMA3	$s+$	$s+$	$s+$	$s+$
LPSO-FCMA2 − LPSO-FCMA3	$s+$	$s+$	$s+$	$s+$
$\tau = 100$, $\rho \Rightarrow$	0.1	1.0	2.0	5.0
LPSO-FCMA1 − LPSO-FCMA2	−	−	+	$s+$
LPSO-FCMA1 − LPSO-FCMA3	$s+$	$s+$	$s+$	$s+$
LPSO-FCMA2 − LPSO-FCMA3	$s+$	$s+$	+	$s+$
$\tau = 200$, $\rho \Rightarrow$	0.1	1.0	2.0	5.0
LPSO-FCMA1 − LPSO-FCMA2	+	+	+	$s+$
LPSO-FCMA1 − LPSO-FCMA3	$s+$	+	$s+$	$s+$
LPSO-FCMA2 − LPSO-FCMA3	$s+$	+	+	+

performs the worst in dynamic environments, which indicates that the random choice scheme is disappointing. Hence, the choice algorithm in LPSO-FCMA2 will always be adopted in the following experiments.

4.4 Experimental Study on the Effect of Diversity Schemes

The main problem for the above investigated PSO-based MAs in dynamic environments lies in that they can loose the exploration capacity for new peaks in the search space due to the convergence of the population. In Sec. 3.3, a SORI approach is extended into PSO for addressing this problem. In this section, we investigate the effect of this SORI scheme on the performance of LPSO-FCMA in dynamic environments via comparing it with another effective diversity scheme, the multi-swarm method, which has been widely used in the literature. In particular, LPSO-FCMA, LPSO-FCMA with the SORI scheme (denoted as LPSO-SOFCMA), multiMPSO with the swarm number set to 5 (denoted as multiMPSO1), and multiMPSO with ten swarms (denoted as multiMPSO2) are applied for DOPs constructed in Table 1.

The population size s_size is set to 70 and the maximal allowable LS size is set to 30 for LPSO-FCMA and LPSO-SOFCMA. For LPSO-SOFCMA, the LS size for the best preserved particle per iteration is always fixed to 5. The population size and the LS size of each sub-swarm are set to 15 and 5 in multiMPSO1, while 5 and 5 in multiMPSO2 respectively. It is easy to understand that multiMPSO1 has a very suitable number of sub-swarms, while multiMPSO2 has a unsuitable number of sub-swarms for the test problems used in this paper. The experimental results are given in Table 4 and plotted in Fig. 9 and the corresponding statistical results are given in Table 5. From Table 4, Table 5 and Fig. 9 several results can be observed and are analyzed as follows.

Firstly, LPSO-SOFCMA always performs better than LPSO-FCMA on all dynamic test problems. LPSO-FCMA can keep a certain population diversity level just depending on its small local neighbor structure, while its particles may converge

Table 4 Experimental results with respect to the overall offline performance of LPSO-FCMAs on the dynamic test problems

Dynamics		Algorithms			
τ	ρ	LPSO-SOFCMA	LPSO-FCMA	multiMPSO1	multiMPSO2
50	0.1	1.81±0.33	2.77±2.61	1.30±0.28	1.84±0.50
50	1.0	2.68±1.24	4.17±5.36	2.21±0.59	4.15±0.81
50	2.0	3.50±1.12	3.68±1.83	2.68±0.51	4.66±0.85
50	5.0	6.32±1.54	7.05±4.24	4.24±0.77	7.94±1.31
100	0.1	1.29±0.99	1.32±1.44	0.64±0.24	1.07±0.19
100	1.0	1.25±0.39	1.32±0.70	1.47±0.26	2.93±0.59
100	2.0	2.03±1.46	2.07±1.31	1.46±0.28	2.69±0.60
100	5.0	3.36±0.72	3.47±0.97	1.97±0.23	3.69±0.55
200	0.1	0.60±0.48	1.22±2.04	0.32±0.06	0.74±0.12
200	1.0	0.79±0.57	0.72±0.38	0.83±0.22	1.84±0.58
200	2.0	1.00±0.82	1.44±1.96	0.70±0.13	1.46±0.54
200	5.0	2.16±2.16	2.11±1.08	1.00±0.15	1.94±0.42

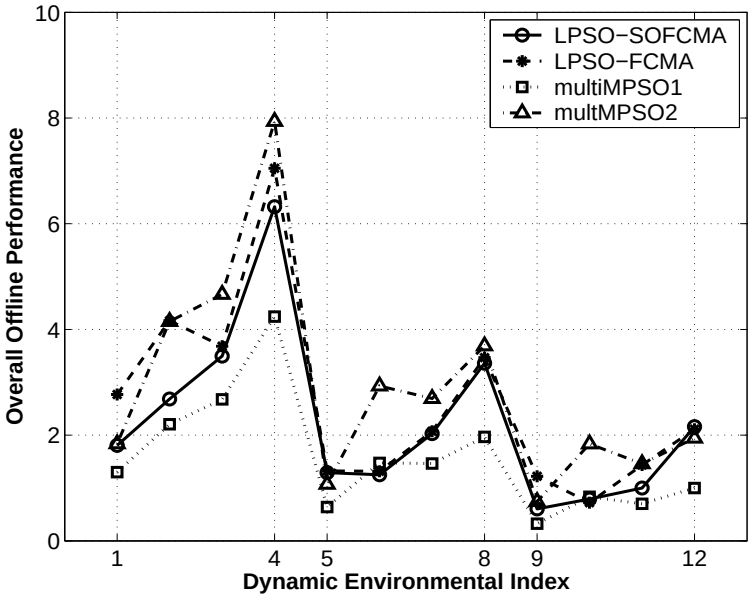

Fig. 9 Experimental results with respect to the overall offline performance of investigated algorithms with different diversity schemes on the dynamic test problems.

into its neighbor best *gbest* as the number of iteration increases. That is, LPSO-FCMA can not achieve a new peak after several iterations. However, this problem can be addressed by the SORI scheme in LPSO-SOFCMA. The SORI scheme can introduce a constant diversity into the population via re-initializing a segment of separate particles randomly every fixed period of iterations. On the other hand, the newly re-initialized particles with low fitness can be preserved and allowed to

Table 5 The t-test results of comparing algorithms regarding the overall offline performance on the dynamic test problems

t-test Result	MPB Problem			
$\tau = 50$, $\rho \Rightarrow$	0.1	1.0	2.0	5.0
LPSO-SOFCMA − LPSO-FCMA	$s+$	$s+$	$+$	$s+$
LPSO-SOFCMA − multiMPSO1	$s-$	$-$	$s-$	$s-$
LPSO-SOFCMA − multiMPSO2	$+$	$s+$	$s+$	$s+$
LPSO-FCMA − multiMPSO1	$s-$	$-$	$s-$	$s-$
LPSO-FCMA − multiMPSO2	$-$	$-$	$s+$	$+$
multiMPSO1 − multiMPSO2	$s+$	$s+$	$s+$	$s+$
$\tau = 100$, $\rho \Rightarrow$	0.1	1.0	2.0	5.0
LPSO-SOFCMA − LPSO-FCMA	$+$	$+$	$+$	$+$
LPSO-SOFCMA − multiMPSO1	$s-$	$s+$	$s-$	$s-$
LPSO-SOFCMA − multiMPSO2	$-$	$s+$	$s+$	$s+$
LPSO-FCMA − multiMPSO1	$s-$	$+$	$s-$	$s-$
LPSO-FCMA − multiMPSO2	$-$	$s+$	$s+$	$+$
multiMPSO1 − multiMPSO2	$s+$	$s+$	$s+$	$s+$
$\tau = 200$, $\rho \Rightarrow$	0.1	1.0	2.0	5.0
LPSO-SOFCMA − LPSO-FCMA	$+$	$-$	$+$	$-$
LPSO-SOFCMA − multiMPSO1	$s-$	$+$	$-$	$s-$
LPSO-SOFCMA − multiMPSO2	$+$	$s+$	$s+$	$-$
LPSO-FCMA − multiMPSO1	$s-$	$+$	$s-$	$s-$
LPSO-FCMA − multiMPSO2	$-$	$s+$	$+$	$-$
multiMPSO1 − multiMPSO2	$s+$	$s+$	$s+$	$s+$

evolve independently for several iterations, which enables LPSO-SOFCMA always keep the exploration capacity for new peaks. The better results of LPSO-SOFCMA over LPSO-FCMA in this experiment indicates that the proper diversity method can improve the performance of algorithms in dynamic environments.

Secondly, multiMPSO1 outperforms LPSO-SOFCMA, while LPSO-SOFCMA performs much better than multiMPSO2 on most dynamic problems. This is because that the performance of multiMPSO for DOPs depends on the number of sub-swarms with a great degree. MultiMPSO can obtain very good results if it owns a proper number of sub-swarms; otherwise, it may perform a little worse. However, the optimal setting regarding the number of sub-swarms is hard to obtain since many dynamic environments are unknown in advance. Thus, the SORI method seems to be a more adaptive diversity scheme for algorithms in dynamic environments. Similar results have also been observed in [29].

4.5 Experimental Study on the Performance of Proposed PSO-Based MAs

In the final set of experiments, we attempt to compare the performance of LPSO-SOFCMA with another existing peer PSO algorithm, multiCPSO algorithm proposed in [2] on the DOPs. Two instances of multiCPSO, multiCPSO with 5 sub-swarm and multiCPSO with 10 sub-swarms (denoted as multiCPSO1 and multiCPSO2 respectively), and a multiMPSO with 5 sub-swarms are considered in the

Table 6 Experimental results with respect to the overall offline performance of LPSO-FCMAs on the dynamic test problems

Dynamics		Algorithms			
τ	ρ	LPSO-SOFCMA	multiMPSO	multiCPSO1	multiCPSO2
50	0.1	1.83±0.38	1.32±0.45	2.31±0.37	2.58±0.34
50	1.0	2.48±0.84	2.17±0.34	2.81±0.58	3.14±0.35
50	2.0	3.16±0.65	2.62±0.51	3.50±0.49	4.52±0.51
50	5.0	6.50±2.05	4.32±1.59	7.13±1.68	10.22±1.94
100	0.1	1.06±0.63	0.61±0.23	1.11±0.14	1.29±0.15
100	1.0	1.25±0.39	1.32±0.70	1.47±0.26	2.93±0.59
100	2.0	1.95±0.89	1.37±0.25	1.68±0.21	2.27±0.33
100	5.0	3.34±0.70	2.17±0.41	3.50±0.47	5.26±0.68
200	0.1	0.59±0.35	0.36±0.16	0.60±0.07	0.68±0.07
200	1.0	0.68±0.33	0.95±0.45	0.66±0.07	0.83±0.09
200	2.0	0.88±0.17	0.74±0.14	0.85±0.12	1.19±0.19
200	5.0	1.66±0.37	1.03±0.17	1.70±0.22	2.61±0.39

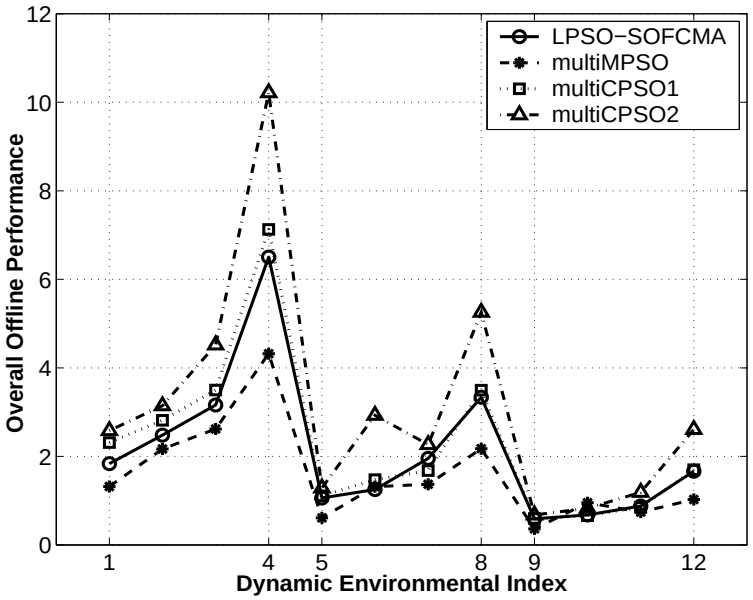

Fig. 10 Experimental results with respect to the overall offline performance of LPSO-SOFCMA and peer algorithms on the dynamic test problems.

experiments. The number of neutral and charged particles in each sub-swarm are set to 10 and 10 for multiCPSO1, while 5 and 5 for multiCPSO2, respectively. The other relevant parameters are always the same as their original settings. For LPSO-SOFCMA and multiMPSO, the same parameter setting in Sec. 4.4 is used again. The experimental results are given in Table 6 and plotted in Fig. 10 and the

Table 7 The t-test results of comparing algorithms regarding the overall offline performance on the dynamic test problems.

t-test Result	Algorithms			
$\tau = 50$, $\rho \Rightarrow$	0.1	1.0	2.0	5.0
LPSO-SOFCMA − multiMPSO	$s-$	$-$	$s-$	$s-$
LPSO-SOFCMA − multiCPSO1	$s+$	$+$	$s+$	$+$
LPSO-SOFCMA − multiCPSO2	$s+$	$s+$	$s+$	$s+$
multiMPSO − multiCPSO1	$s+$	$s+$	$s+$	$s+$
multiMPSO − multiCPSO2	$s+$	$s+$	$s+$	$s+$
multiCPSO1 − multiCPSO2	$s+$	$s+$	$s+$	$s+$
$\tau = 100$, $\rho \Rightarrow$	0.1	1.0	2.0	5.0
LPSO-SOFCMA − multiMPSO	$s-$	$+$	$s-$	$s-$
LPSO-SOFCMA − multiCPSO1	$+$	$s+$	$-$	$+$
LPSO-SOFCMA − multiCPSO2	$s+$	$s+$	$+$	$s+$
multiMPSO − multiCPSO1	$s+$	$+$	$s+$	$s+$
multiMPSO − multiCPSO2	$s+$	$s+$	$s+$	$s+$
multiCPSO1 − multiCPSO2	$s+$	$s+$	$s+$	$s+$
$\tau = 200$, $\rho \Rightarrow$	0.1	1.0	2.0	5.0
LPSO-SOFCMA − multiMPSO	$s-$	$s+$	$s-$	$s-$
LPSO-SOFCMA − multiCPSO1	$+$	$-$	$-$	$+$
LPSO-SOFCMA − multiCPSO2	$+$	$s+$	$s+$	$s+$
multiMPSO − multiCPSO1	$s+$	$s-$	$s+$	$s+$
multiMPSO − multiCPSO2	$s+$	$-$	$s+$	$s+$
multiCPSO1 − multiCPSO2	$s+$	$s+$	$s+$	$s+$

corresponding statistical results are given in Table 7. From Table 6, Table 7, and Fig. 10, several results can be observed and are analyzed as follows.

Firstly, LPSO-SOFCMA always outperforms multiCPSO1 on the dynamic problems when the environment changes quickly, i.e., when $\tau = 50$. When $\tau = 50$, LPSO-SOFCMA can always achieve the optimum more quickly than multiCPSO1 because the FCLS operator has a strong exploitation capacity. The similar conclusion can be further drawn from the t-test results regarding multiMPSO − multiCPSO in Table 7, which will be explained in the later experimental analysis. When $\tau = 100$ or 200, the situation becomes a little different. Two algorithms exhibit almost the same performance, though LPSO-SOFCMA performs a little better than multiCPSO1 on dynamic problems with $\rho = 0.1$ or 5.0, while multiCPSO1 performs a little better than LPSO-SOFCMA when $\rho = 1.0$ or 2.0.

Secondly, multiMPSO significantly outperforms multiCPSO1 on most dynamic environments, which is a very interesting result. The only difference between the two algorithms multiMPSO and multiCPSO1 lies in the update course of sub-swarms. In multiMPSO, the best fitness particle of each sub-swarm is improved by the FCLS operator, while the update of charged particles in multiCPSO1 can be affected by the exclusion force among them. Obviously, the FCLS operator in multiMPSO and the charged particles in multiCPSO1 are both used to ensure the algorithm adapt to the environmental changes quickly. The good performance of multiMPSO over multiCPSO1 shows that our proposed FCLS operator has a very strong robustness and adaptivity in dynamic environments.

Thirdly, the behavior of multiMPSO2 seems to be disappointing and it is almost beaten by the other algorithms on all dynamic environments due to its improper number of sub-swarms, which has been explained in the above experimental analysis (see Sec. 4.4).

Finally, the environmental parameters affect the performance of algorithms. The performance of all algorithms increases when the value of τ increase from 50 to 100 to 200 or when the value of ρ decreases. It is easy to understand because when τ becomes larger, algorithms have more time to find better solutions before the next change, while the new optimum is closer to the previous one when ρ becomes smaller.

5 Conclusions

In this paper, a PSO-based memetic algorithm is proposed and experimentally investigated in dynamic environments. In the proposed memetic algorithm, a local version of PSO with a ring-shape topology structure is hybridized with a fuzzy cognition local search method, which is proposed based on a fuzzy cognition-only model of PSO. To further enhance the exploration capacity of algorithm, a self-organized random immigrants scheme is also extended into our proposed memetic algorithm for DOPs. From the experimental results, the following conclusions can be drawn on the dynamic test problems.

First, a local version of PSO algorithm enhanced by some suitable techniques can exhibit a better performance in dynamic environments. For most dynamic problems, LPSO-SOFCMA, which hybridized LPSO with the FCLS operator and the SORI scheme, outperforms most of other peer algorithms.

Second, the FCLS operator can help MAs execute a robust local refinement in both stationary and dynamic environments. In the experiments, LPSO-FCMA always performs better than LPSO-RWMA for the stationary test problem, while multiMPSO always outperforms multiCPSO for the dynamic test problems.

Third, it is always beneficial that several high fitness particles within different peaks are selected from the population for local improvements when MAs are applied in dynamic multi-modal environments. The choice algorithm proposed in this paper proves to be a good solution for this problem.

Fourth, the diversity scheme can be efficient for improving the performance of algorithms for DOPs. Comparing with the multi-swarm method, SORI seems to be a more adaptive diversity scheme for algorithms in dynamic environments.

Finally, the environmental dynamics can affect the performance of algorithms. In our experiments, algorithms perform better with the increasing of the frequency of changes, while with the decreasing of the severity of changes.

Generally speaking, the experimental results indicate that the proposed algorithm, where a local version PSO is hybridized with the FCLS operator and the SORI scheme, seems a good optimizer for dynamic optimization problems.

For the future work, it is straightforward to compare our proposed PSO-based MAs with some existing state-of-the-art PSOs for DOPs, such as the species-based PSO in [24], multi-swarm quantum swarm optimization in [2], etc., in order to further examine the performance of the proposed algorithm in dynamic environments. Another interesting research work is that some other diversity schemes, such as

memory-based mechanisms and speciation approaches, can be considered to be hybridized with the proposed MAs for DOPs. In addition, it is also valuable to carry out the sensitivity analysis on the effect of parameters, e.g., r_s, r_r, and t_ri, on the performance of the PSO-based MAs for DOPs in the future.

Acknowledgments

We would like to thank the anonymous associate editor and reviewers for their thoughtful suggestions and constructive comments. We would also like to thank Dr. Il-Kyeong Moon from Pusan National University, Korea, for all the discussions regarding the paper. The work by Hongfeng Wang was supported by the National Nature Science Foundation of China (NSFC) under Grant No. 70431003 and Grant No. 70671020, the National Innovation Research Community Science Foundation of China under Grant No. 60521003, the National Support Plan of China under Grant No. 2006BAH02A09 and the Ministry of Education, science, and Technology in Korea through the Second-Phase of Brain Korea 21 Project in 2009. The work by Shengxiang Yang was supported by the Engineering and Physical Sciences Research Council (EPSRC) of UK under Grant EP/E060722/01. The work by W. H. Ip was supported by the Hong Kong Polytechnic University Research Grants under Grant G-YH60.

References

1. T. M. Blackwell and P. Bentley (2002). Dynamic search with charged swarms. *Proc. of the 2002 Genetic and Evol. Comput. Conf.*, pp. 19-26.
2. T. M. Blackwell and J. Branke (2006). Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Trans. on Evolutionary Computation*, **10**(4): 459-472.
3. 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.
4. J. Branke. The moving peaks benchmark website. Online, http://www.aifb.unikarlsruhe.de/jbr/MovPeaks.
5. J. Branke (1999). Memory enhanced evolutionary algorithms for changing optimization problems. *Proc. of the 1999 Congress on Evolutionary Computation*, pp. 1875-1882.
6. A. Carlisle and G. Dozier. Adapting particle swarm optimization to dynamic environments. *Proc. of the 2000 Int. Conf. on Artificial Intelligence*, pp. 429-434.
7. 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.
8. R. C. Eberhart and Y. Shi (2001). Tracking and optimizing dynamic systems with particle swarms. *Proc. of the 2001 IEEE Congress on Evolutionary Computation*, pp. 94-97.
9. J. E. Gallardo, C. Cotta and A. J. Ferandez (2007). On the hybridization of memetic algorithms with branch-and-bound Techniques. *IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics*, 37(1): 77-83.
10. J. J. Grefenstette (1992). Genetic algorithms for changing environments. *Proc. of the 2nd Int. Conf. on Parallel Problem Solving from Nature*, pp. 137-144.

11. C. K. Goh and K. C. Tan (2009). A competitive-cooperation coevolutionary paradigm for dynamic multi-objective optimization. *IEEE Trans. on Evol. Comput.*, 13(1): 103-127.
12. X. Hu and R. Eberhart (2002). Adaptive particle swarm optimization: Detection and response to dynamic systems. *Proc. of the 2002 IEEE Congress on Evolutionary Computation*, pp. 1666-1670.
13. I. Hatzakis and D. Wallace (2006). Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach. *Proc. of the 2006 Genetic and Evol. Comput. Conference*, pp. 1201-1208.
14. H. Ishibuchi, T. Yoshida and T. Murata (2003). Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Trans. on Evol. Comput.*, 7(2): 204-223.
15. Y. Jin and J. Branke (2005). Evolutionary pptimization in uncertain environments-a survey. *IEEE Trans. on Evol. Comput.*, 9(3): 303-317.
16. S. Janson and M. Middendorf (2006). A hierarchical particle swarm optimizer for noisy and dynamic environments. *Genet Program Evolvable Mach*, 7(3): 329-354.
17. J. Kennedy (1997). The Particle Swarm: Social Adaptation of Knowledge. *Proc. of IEEE International Conference on Evolutionary Computation*, pp. 303-308.
18. N. Krasnogor and J. E. Smith (2005). A tutorial for competent memetic algorithms: model, taxonomy and design issues. *IEEE Trans. on Evol. Comput.*, 9(5): 474-488.
19. X. Li and K. H. Dam (2003). Comparing particles swarms for tracking extrema in dynamic environments *Proc. of the IEEE 2003 Congress on Evol. Comput.*, pp. 1772-1779.
20. D. Liu, K. C. Tan, C. K. Goh, and W. K. Ho (2007). A multiobjective memetic algorithm based on particle swarm optimization. *IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics*, 37(1): 42-50.
21. B. Liu, L. Wang and Y. H. Jin (2007). An effective PSO-based memetic algorithm for flow shop scheduling. *IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics*, 37(1): 18-27.
22. L. Liu, S. Yang and D. Wang (2008). Compound Particle Swarm Optimization in Dynamic Environments. *Proc. of EvoWorkshops 2008*, pp. 616-625.
23. S. Man, Y. Liang, K. S. Leung, K. H. Lee and T. S. K. Mok (2007). A memetic algorithm for multiple-drug cancer chemotherapy schedule optimization. *IEEE Trans. on Systems, Man and Cybernetics-Part B: Cybernetics*, 37(1): 84-91.
24. 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.
25. Y. G. Petalas, K. E. Parsopoulos and M. N. Vrahatis (2007). Memetic particle swarm optimization. *Ann Oper Res*, 156: 99-127.
26. J. E. Smith (2007). Coevolving memetic algorithms: A review and progress report, *IEEE Trans. on SMC-Part B: Cybern.*, 37(1): 6-17.
27. J. Tang, M. H. Lim and Y.-S. Ong (2007). Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems. *Soft Computing*, 11(10): 957-971.
28. M. Tang and X. Yao (2007). A memetic algorithm for VLSI floorplanning. *IEEE Trans. on Systems, Man and Cybernetics-Part B: Cybernetics*, 37(1): 62-69.
29. R. Tinos and S. Yang (2007). A self-organizing random immigrants genetic algorithm for dynamic optimization problems. *Genet Program Evolvable Mach*, 8(3): 255-286.
30. A. S. Uyar and A. E. Harmanci. A new population based adaptive dominance change mechanism for diploid genetic algorithms in dynamic environments. *Soft Computing*, 9(11): 803-815, 2005.
31. F. van den Bergh (2002). An analysis of particle swarm optimizers. *PhD thesis*, University of Pretoria, South Africa, 2002.

32. H. Wang and D. Wang (2006). An improved primal-dual genetic algorithm for optimization in dynamic environments. *Proc. of the 13th Int. Conf. on Neural Information Processing*, Part III, pp. 836-844.
33. H. Wang, D. Wang and S. Yang (2007). Triggered memory-based swarm optimization in dynamic environments. *Proc. of the EvoWorkshop 2007*, pp. 637-646.
34. 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.*, to appear.
35. H. Wang, S. Yang and D. Wang (2009). A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems. *Soft Computing*, 13(8-9): 763-780.
36. S. Yang (2003). Non-Stationary Problem Optimization Using the Primal-Dual Genetic Algorithm. *Proc. of the 2003 Congress on Evolutionary Computation*, vol. 3, pp. 2246-2253.
37. S. Yang (2008). Genetic algorithms with memory and elitism based immigrants in dynamic environments. *Evolutionary Computation*, 16(3): 385-416.
38. S. Yang, Y.-S. Ong, and Y. Jin (eds.), *Evolutionary Computation in Dynamic and Uncertain Environments*, Springer-Verlag, Berlin Heidelberg, 2007.
39. S. Yang and X. Yao (2005). Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Computing*, 9(11): 815-834.
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.