Applied Soft Computing 32 (2015) 432–448

Contents lists available at ScienceDirect

Applied Soft Computing

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

A novel hybrid adaptive collaborative approach based on particle swarm optimization and local search for dynamic optimization problems

Ali Sharifi[a], Javidan Kazemi Kordestani[b,*], Mahshid Mahdaviani[a], Mohammad Reza Meybodi[a]

[a] Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), 424 Hafez Ave., Tehran, Iran
[b] Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

ARTICLE INFO

Article history:
Received 6 August 2014
Received in revised form 1 January 2015
Accepted 2 April 2015
Available online 13 April 2015

Keywords:
Dynamic optimization problems
Moving peaks benchmark
DOPs
MPB
Particle swarm optimizer
Naive direct search

ABSTRACT

This paper proposes a novel hybrid approach based on particle swarm optimization and local search, named PSOLS, for dynamic optimization problems. In the proposed approach, a swarm of particles with fuzzy social-only model is frequently applied to estimate the location of the peaks in the problem landscape. Upon convergence of the swarm to previously undetected positions in the search space, a local search agent (LSA) is created to exploit the respective region. Moreover, a density control mechanism is introduced to prevent too many LSAs crowding in the search space. Three adaptations to the basic approach are then proposed to manage the function evaluations in the way that are mostly allocated to the most promising areas of the search space. The first adapted algorithm, called HPSOLS, is aimed at improving PSOLS by stopping the local search in LSAs that are not contributing much to the search process. The second adapted, algorithm called CPSOLS, is a competitive algorithm which allocates extra function evaluations to the best performing LSA. The third adapted algorithm, called CHPSOLS, combines the fundamental ideas of HPSOLS and CPSOLS in a single algorithm. An extensive set of experiments is conducted on a variety of dynamic environments, generated by the moving peaks benchmark, to evaluate the performance of the proposed approach. Results are also compared with those of other state-of-the-art algorithms from the literature. The experimental results indicate the superiority of the proposed approach.

1. Introduction

Optimization in dynamic environments has emerged as an important field of research during the last two decades, since many real-world optimization problems tend to change over time. The representative examples of real-world dynamic optimization problems (DOPs) are portfolio decisions optimization in changing stock market conditions, shortest path routing problem in changing network environments, and vehicle routing with online arrival of customer's requests. In these problems, the task of optimization is not confined to finding the optimum solution(s) in the problem space but to continuously and accurately adapt to the new conditions during the course of optimization.

Each dynamic optimization problem P can be defined by a quintuple $\{\Omega, x, \phi, f, t\}$ where Ω denotes the search space, x is a feasible solution in Ω, ϕ represents the system control parameters which determines the distribution of the solutions in the fitness landscape, f is the static objective function and t is the time. P is then can be modeled as follows [1]:

$$P = \sum_{t=0}^{\text{end}} f_t(x, \phi) \quad (1)$$

As can be seen in the above equation, dynamic optimization problem P is composed of a sequence of static instances. Hence, the goal of the optimization in such problems is no longer just locating the optimal solution(s), but rather tracking the shifting optima over the time.

* Corresponding author. Tel.: +98 9373908326.
E-mail addresses: en.sharifi.ali@gmail.com (A. Sharifi), javidan.kazemi@gmail.com (J. Kazemi Kordestani), mahshid.mahdaviani@gmail.com (M. Mahdaviani), mmeybodi@aut.ac.ir (M.R. Meybodi).

http://dx.doi.org/10.1016/j.asoc.2015.04.001
1568-4946/© 2015 Elsevier B.V. All rights reserved.

The dynamism of the problem can then be obtained by tuning the system control parameters as follows:

$$\phi_{t+1} = \phi_t \oplus \Delta\phi \tag{2}$$

where $\Delta\phi$ is the deviation of the control parameters from their current values and $\oplus$ represents the way the parameters are changed. The next state of the environment then can be defined using current state of the environment as follows:

$$f_{t+1}(x, \phi) = f_t(x, \phi_t \oplus \Delta\phi) \tag{3}$$

Different change types can be defined, using $\Delta\phi$ and $\oplus$. Li et al. [1] proposed a framework of the eight change types including *small step change*, *large step change*, *random change*, *chaotic change*, *recurrent change*, *recurrent change with noise*, *and dimensional change*. For additional information, interested readers are referred to [1].

The adaptive nature of evolutionary algorithms and swarm intelligence methods make them suitable candidates for dealing with DOPs. However, the dynamic behavior of DOPs poses additional challenges to the evolutionary algorithms and swarm intelligence techniques. The main challenge of the mentioned techniques in dynamic environments is *diversity loss* which arises due to the tendency of the population to converge to a single optimum. Consequently, when a change occurs in environment, the number of function evaluations required for a partially converged population to re-diversify and re-converge to the shifted optimum is quite deleterious to the performance [2]. Particle swarm optimization (PSO) is one of the swarm intelligence algorithms that should undergo a certain adjustment to work well in dynamic environments.

In this paper, a novel PSO-based approach which combines a *fuzzy social-only model* PSO and local search is suggested to address DOPs. The basic algorithm is then further extended based on three resource management schemes. Experimental studies are carried out on the sensitivity analysis of the several components and parameters of the proposed method. The performance of the proposed approach is also compared with several state-of-the-art algorithms on moving peaks benchmark (MPB) problem.

The rest of this paper is structured as follows: Section 2 gives an overview to the basic PSO algorithm and its variants for dynamic environments. In Section 3, we explain our proposed approach in detail. Experimental study regarding the parameters sensitivity analysis, impact of using different local search methods, effect of applying different resource management schemes, and comparison with other state-of-the-art methods is presented in Section 4. Finally, conclusions and future works are given in Section 5.

2. Background and related works

PSO is a versatile population-based stochastic optimization method which was first proposed by Kennedy and Eberhart [3] in 1995. PSO begins with a population of randomly generated particles in a *D*-dimensional search space. Each particle *i* of the swarm has three features: $\vec{x}_i$ that shows the current position of the particle *i* in the search space, $\vec{v}_i$ which is the velocity of the particle *i* and $\vec{p}_i$ which denotes the best position found so far by the particle *i*. Each particle *i* updates its position in the search space, at every time step *t*, according to the following equations:

$$\vec{v}_i(t+1) = w\vec{v}_i(t) + c_1r_1[\vec{p}_i(t) - \vec{x}_i(t)] + c_2r_2[\vec{p}_g(t) - \vec{x}_i(t)] \tag{4}$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \tag{5}$$

where w is the inertia weight parameter which determines the portion of velocity preserved during the last time step of the algorithm. c_1 and c_2 denote the cognitive and social learning factors that are used to adjust the degree of the movement of particles toward their personal best position and global best position of the swarm, respectively. r_1 and r_2 are two independent random variables drawn with uniform probability from [0,1]. Finally, $\vec{p}_g$ is the globally best position found so far by the swarm. The pseudo-code of the PSO is shown in Algorithm 1.

Algorithm 1. Pseudo-code for canonical PSO

```
1.  Setting parameters
2.  Generate the initial swarm of the particles with random positions and velocities
3.  Evaluate the fitness of each particle of the swarm
4.  repeat
5.      for each particle i in the swarm do
6.          update particle i according to Eqs.(4),(5);
7.          if(fitness($\vec{p}_i$)<fitness($\vec{x}_i$)) then
8.              $\vec{p}_i := \vec{x}_i$;
9.              if(fitness($\vec{p}_g$)<fitness($\vec{x}_i$)) then
10.                 $\vec{p}_g := \vec{x}_i$;
11.             end-if
12.         end-if
13.     end-for
14. until a termination condition is met
```

PSO has been successfully applied in many optimization problems including numerical optimization [4,5], image processing [6], training feedforward neural networks in dynamic environments [7], density estimation [8], multi-objective optimization [9,10], data clustering [11], etc. However, as we mentioned in Section 1, PSO suffers from *diversity loss* when dealing with DOPs. Many attempts have been made by the researchers to deal with this issue. In the rest of this section, we will examine the current studies and advances in the literature in five major groups.

2.1. Injecting diversity to the swarm after detecting a change in the environment

A very basic strategy to cope with *diversity loss* problem is to inject a certain level of diversity to the swarm whenever a change occurs in the environment. An example of this approach is the re-randomization PSO (RPSO) proposed by Hu and Eberhart [12]. In RPSO, upon detecting a change in the environment, the entire or a part of the swarm is randomly relocated in the search space. Their results suggest that the re-randomization of a small portion of the swarm (i.e. 10% in their study) can improve the ability of the canonical PSO in tracking the shifting optimum, when the step size of the environmental change is small. It should be noted that specifying the proper rate of randomization is a challenging task. One the one hand, too much randomization will result in losing too much information and resemble solving the problem from scratch. One the other hand, too little randomization might be unable to introduce enough diversity to the swarm, which is necessary for tracking the optimum.

2.2. *Maintaining diversity throughout the run*

A group of studies have tried to preserve a sustainable level of diversity during the entire run by avoiding individuals from converging to a single optimum.

2.2.1. *Mutual repulsion*

Applying a mutual repulsion between particles, swarms or captured optimums can maintain a suitable level of diversity during the search process. The atomic model [13,14] used in mCPSO is an example of utilizing repulsion for dynamic environments. In mCPSO, each swarm contains a number of *charged classical* particles. Charged classical particles in each swarm repel each other to provide a sustainable diversity among the individuals.

2.2.2. *Dynamic information topology*

Some researchers have modified the topology of information sharing among the particles of the swarm in order to improve the performance of PSO in dealing with dynamic situations. This can temporarily reduce the tendency of particles to move toward the best found positions, thereby enhancing the diversity among the swarm. For example, Janson and Middendorf [15] studied a hierarchical PSO (H-PSO) for noisy and dynamic environments. H-PSO uses a tree-like structure to establish a hierarchy between particles. They also proposed a partitioned hierarchical PSO (PH-PSO) where the tree is temporarily partitioned into sub-swarms after an environmental change is detected. Both approaches reported a significant improvement over standard PSO on different dynamic and noisy functions.

2.2.3. *Others*

Beside the above strategies, some researchers have considered other types of techniques to maintain diversity throughout the run. Li and Yang [16], for example, incorporated a *random immigrants* scheme into PSO where a temporal population is randomly generated if the population diversity drops below a constant threshold.

2.3. *Memory-based approaches*

Many studies have proved that making use of a memory scheme to restore useful information about previously found positions in the search space can be beneficial for various DOPs. For example, Wang et al. [17] proposed a memory-based PSO for dynamic environments. In this method, the entire population is divided into three parts: (1) “explore”-population which is responsible for collecting information about the good solutions, (2) “memory”-population for storing and retrieving information about promising areas of the search space, and (3) “exploit”-population that is used to refine the solutions around the areas provided by memory. They have introduced two new operators called *memory-based resetting* and *memory-based immigrants* to transfer information from memory to “exploit”-population. Their experimental results confirm that the memory scheme can enhance the performance of PSO in dynamic environments.

2.4. *Multi-swarm PSO*

The most widely used technique to cope with *diversity loss* problem is to make use of multiple swarms or to divide the particles of the main swarm into several sub-swarms, and assign them to different areas of the search space. In the rest of this sub-section, we examine two major types of multi-swarm PSOs for DOPs.

2.4.1. *Methods with a fix number of swarms*

Various methods exist in the literature that have used a fixed number of swarms to capture several promising areas of the search space in parallel. The pioneer work in this context was done by Blackwell and Branke [18]. They introduced a multi-swarm algorithm based on the atomic metaphor, namely mQSO. The mQSO starts with a predefined number of swarms, typically equal to the number of peaks in the landscape, exploring in the search space to locate the optima. In mQSO, each swarm is composed of a number of *neutral* particles and a number of *quantum* particles. Neutral particles update their velocity and position according to the principles of pure PSO. On the other hand, quantum particles do not move according to the PSO dynamics but change their positions around the center of the best particle of the swarm according to a random uniform distribution with radius r_{cloud}. Consequently, they never converge, but provide the swarm with a sustainable level of diversity needed for tracking the shifting optimum. They also introduced two operators to establish two forms of interactions between swarms. First operator named *exclusion*, which prevents populations from settling on the same peak by reinitializing the worst performing swarm in the search space. Second operator is referred to as *anti-convergence* which is triggered whenever all swarms converge, and removes the worst swarm from its peak and reinitializes it in the search space. These two operators have initiated two implications for other researchers as: (a) search of two populations in the same area is not efficient and (b) it is necessary for different methods to be able to detect new emerging peaks, specifically in the environments with an unknown number of peaks.

2.4.2. *Methods with varying number of swarms*

2.4.2.1. Use of adaptation mechanisms for determining the number of populations. The critical drawback of the multi-swarm algorithm proposed in [18] is that the number of swarms should be defined before the optimization process, whereas in real-world problems the information about the environment may not be in hand. The very first attempt to adapt the number of swarms in dynamic environments was made by Blackwell [2]. He introduced self-adaptive version of mQSO, which dynamically adjusts the number of swarms either by spawning new swarms into the search space, or by destroying redundant swarms [2]. In this algorithm, swarms are categorized into two groups: *free* swarms, whose expansion, i.e. the maximum distance between any two particles in the swarm in all dimensions, are larger than radius r_{conv}, and *converged* swarms. Once the expansion of a *free* swarm, becomes smaller than a radius r_{conv}, it is converted to the *converged* swarm. In AmQSO, when the number of *free* swarms (M_{free}) is dropped to zero, a *free* swarm is initialized in the search space for capturing undetected peaks. On the other hand, *free* swarms are removed from the search space if M_{free} is higher than a threshold n_{excess}.

2.4.2.2. Clustering the main swarm into some sub-swarms. Dividing the main population of the algorithm into several clusters, i.e. sub-populations, via clustering methods and assigning each of them to different promising regions of the search space is a relatively new multi-population scheme for DOPs which has shown promising results [19]. For example, Parrott and Li [20] proposed a Speciation-based Particle Swarm Optimization (SPSO) for tracking multiple optima in dynamic environments, which dynamically distributes particles of the main swarm over a variable number of so-called *species*.

Yang and Li [21] employed a single linkage hierarchical clustering method to create an adaptive number of sub-swarms, and assign them to different promising sub-regions of the search space. Each created sub-swarm then uses the PSO with a modified *gbest* model to exploit the respective region. In order to avoid different clusters from crowding, they have applied a redundancy control check. In this regard, when two sub-swarms are located on a single peak, first they are merged together and then the worst performing individuals of the sub-swarm are removed until its size is equal to a predefined threshold. In another work [16], they have further extended the fundamental idea of CPSO and introduced a framework for multi-population methods in undetectable environments, where the detection of environmental changes is very difficult or impossible.

Nickabadi et al. [22] proposed a competitive clustering particle swarm optimizer (CCPSO) for DOPs. They employed a multi-stage clustering procedure to split the particles of the main swarm over a varying number of sub-swarms based on the particles positions as well as their objective function values. In addition to the sub-swarms, there is also a group of free particles that explore the environment to locate new emerging optima or exploit the current optima which are not followed by any sub-swarm, and are updated using a cognitive-only movement model. Moreover, each sub-swarm adjusts its inertia weight according to the success percentage of the sub-swarm [23].

2.4.2.3. Multi-swarm scheme based on space partitioning. Another approach for dealing with *diversity loss* problem is to divide the search space into several partitions and keep the number of individuals in each partition less than a predefined threshold. Hashemi and Meybodi [24] incorporated PSO and Cellular Automata (CA) into an algorithm referred to as cellular PSO. In cellular PSO, the search space is partitioned into some equally sized cells using CA. Then, particles of the swarm are allocated to different cells according to their positions in the search space. At any time, particles residing in each cell use their personal best experience and the best solution found in their neighborhood cells for searching an optimum. Moreover, whenever the number of particles within each cell exceeds a predefined threshold, randomly selected particles from the saturated cells are transferred to random cells within the search space. In addition, each cell has a memory that is used to keep track of the best position found within the boundary of the cell and the best position among its neighbors. In another work [25], they changed the role of some particles in each cell, from standard particles to quantum particles for a few iterations upon the detection of a change in the environment.

With the aim to improve the performance of cellular PSO, Sharifi et al. [26] proposed a two phased collaborative algorithm, named two phased cellular PSO (TP-CPSO). In TP-CPSO, each cell can function in two different phases: *exploration* phase, in which the search is performed by a modified PSO algorithm until a convergence to local optima is detected, and *exploitation* phase, in which the search process is conducted by a directed local search with a high capability of exploitation. Initially, all cells are in *exploration* phase, but upon converging to local optima they go to *exploitation* phase. Moreover, they eliminated the need for defining a limit for each cell.

2.4.2.4. Methods with a parent swarm and various child swarms. The major strategy in this approach is to divide the search space into different sub-regions, using a parent swarm, and carefully exploit each sub-region with a distinct child swarm. Several studies exist in the literature that have followed this approach. For example, Kamosi et al. [27] introduced a multi-swarm algorithm for dynamic environments, which consists of a parent swarm to explore the search space and a varying number of child swarms to exploit promising areas found by the parent swarm. In this method, called mPSO, the optimization process starts with a parent swarm exploring the entire search space for locating promising areas of the search space. Whenever the fitness value of the best particle in the parent swarm improves, a child swarm will be created as follows: (a) a number of particles located within the radius r from the best particle of the parent swarm are separated from the parent swarm and form a new child swarm and (b) a number of particles are randomly initialized in a hyper-sphere with radius $r/3$ centered at the best particle of the child swarm. Afterwards, a number of randomly initialized particles are added to the parent swarm in order to compensate for the migration of particles from parent swarm to the child swarm. After creating child swarms, each of them exploits their respective sub-regions in the search space. Upon detecting a collision between two child swarms, the worse child swarm is removed. In case that a particle of the parent swarm finds a better position in the search territory of a child swarm, it replaces the local best particle of the respective child swarm. Afterwards, the particle of the parent swarm is reinitialized in the search space. In another work, inspired by hibernation phenomenon in animals, they extended mPSO and proposed a hibernating multi-swarm algorithm (HmSO) [28]. In HmSO, unproductive search in child swarms is stopped by "sleeping" converged child swarms. This hibernation mechanism makes more function evaluations available for the parent swarm to explore the search space, and for the other child swarms to exploit their respective areas in the landscape.

Recently, Yazdani et al. [29] employed several mechanisms in a multi-swarm algorithm to conquer the existing challenges in dynamic environments. In their proposed method, named FTMPSO, a randomly initialized *finder* swarm exploring the entire search space for locating the position of the peaks. Upon convergence of the finder swarm, a *tracker* swarm will be activated by transferring the fittest particles from the *finder* swarm into the newly created *tracker* swarm. The *finder* swarm is then reinitialized into the search space to capture other uncovered peaks. Afterwards, the activated *tracker* swarm is responsible for finding the top of the peak using a local search along with following the respective peak upon detecting a change in the environment. In order to avoid *tracker* swarms exploiting the same peak, *exclusion* is applied between every pair of *tracker* swarms. Besides, if the *finder* swarm converges to a populated peak, it is reinitialized into the search space without generating any *tracker* swarm.

Our proposed approach is relatively similar to the methods of this group in the sense that it uses a main swarm to explore the search space. However, instead of using child swarms for exploiting promising areas of the search space, in this work we apply local search agents. Moreover, we use various schemes to effectively allocate the function evaluations to the most promising areas of the search space.

2.5. PSO with adaptive scheme

In this approach, knowledge on the search process from different particles is included to improve the efficiency and effectiveness of the search process. Moreover, in this approach, different schemes and operators are further devised to promote the ability of the particles to deal with the environmental dynamism. For example, inspired by composite particles in physic, Liu et al. [30] introduced a new variant of PSO, called PSO with composite particles (PSO-CP) for DOPs. PSO-CP contains four major components: (1) composite particles which are formed by iteratively grouping the worst particles with two of their nearest particles, (2) scattering operator which aims at enhancing the local diversity within composite particles, (3) velocity-anisotropic reflection scheme that can improve the ability of particles for tracking the promising peaks in a new environment, and (4) integral movement strategy which is favorable for detecting new peaks in the search space by promoting the swarm diversity. Inspired by microbial life, Karimi et al. [31] proposed the dynamic hybrid PSO (DHPSO) where a new birth and death mechanism has incorporated into the PSO to dynamically adjust the swarm size during the search process. In DHPSO, each three neighborhood particles can produce a new infant, based on the quadratic interpolation distribution, with a certain probability. Moreover, they have used an aging scheme to remove the old particles.

2.6. Hybrid methods

Some researchers have tried to combine desirable features of PSO with other optimization algorithms for DOPs. For example, Lung and Dumitrescu [32] introduced a hybrid collaborative approach for dynamic environments called collaborative evolutionary swarm optimization (CESO). CESO has two equal-size populations: a main population to maintain a set of local and global optima during the search process using crowding-based differential evolution (CDE), and a PSO population acting as a local search operator around solutions provided by the first population. During the search process, information is transmitted between both populations via collaboration mechanisms. In another work [33], a third population is incorporated to CESO which acts as a memory to recall some promising information from past generations of the algorithm. Inspired by CESO, Kordestani et al. [34] proposed a bi-population hybrid algorithm. First population, called QUEST, is evolved by CDE principles to locate the promising regions of the search space. Second population, called TARGET, uses PSO to exploit useful information in the vicinity of the best position found by the QUEST. When the search process of the TARGET population around the best found position of the QUEST becomes unproductive, TARGET population is stopped and all genomes of the QUEST population are allowed to perform extra operation using hill-climbing local search. They also applied several mechanisms to conquer the existing challenges in the dynamic environments. In this work, the authors have tried to spend more function evaluations around best found position in the search space. Here, our adaptations are also following the idea of spending more function evaluations around the best found position in the search space.

3. Proposed algorithms

In this section, first we describe the outline of the basic approach in detail. Three adaptations to the basic approach are then proposed to further enhance the performance of the basic algorithm by managing the function evaluations.

3.1. Basic approach

3.1.1. General procedure

PSOLS stars with a randomly initialized swarm exploring the entire search space to locate promising areas using a social-only movement model. Upon detecting such areas, a local search agent is created and placed on the best position found by the swarm. The swarm is then reinitialized into the search space to explore for locating other desirable regions. After creating LSAs, each of them is responsible for exploiting its respective region and tracking the respective peak after the change. Once an environmental change is detected, the fitness of all LSAs are re-evaluated and their step size is adjusted to the initial value. Moreover, the swarm is reinitialized. The structure of PSOLS is given in Algorithm 2. In the rest of this sub-section, the major steps of PSOledLS are described in detail.

3.1.2. Initialization

The initial swarm of particles is simply randomized into the boundary of the search space according to a uniform distribution as follows:

$$x_{ij} = L_j + rand[0, 1] \times (U_j - L_j) \tag{6}$$

where $i \in [1, 2, \ldots, swarm\_size]$ is the index of ith particle of the swarm, $j \in [1, 2, \ldots, D]$ represents jth dimension of the search space, *rand* [0,1] is a uniformly distributed random number in the range of [0,1]. Finally, L_j and U_j are the lower and upper bounds of the search space corresponding to jth dimension.

Moreover, the initial velocity of each particle is defined according to the following equation:

$$v_{ij} = (1 - 2 \times rand[0, 1]) \times (v_{max}) \tag{7}$$

where v_{ij} is the velocity of ith particle in jth dimension of the search space, *rand* [0,1] is again a random uniform number in [0,1], and v_{max} is the maximum velocity.

Algorithm 2. General Procedure of PSOLS

```
1.  Create an empty list lsas to record local search agents;
2.  Generate the initial swarm of the particles with random positions and velocities according to Eqs. (6) and (7);
3.  Evaluate the fitness of each particle;
4.  repeat
5.      if a change is detected in the environment then
6.          Reinitialize swarm;
7.          Re-evaluate all LSAs in lsas;
8.          Adjust the step size for LSAs;
9.      end-if
10.     for each particle i in the swarm do
11.         Update particle i according to Eqs. (10), (8) and (9), respectively;
12.     end-for
13.     if hasConverged(swarm) then
14.         if there is no LSA in lsas in the radius $\xi$ from $\vec{p}_g$ then
15.             Add a new LSA with location $\vec{p}_g$ to lsas;
16.         else
17.             densityControl(lsas);
18.         end-if
19.         Reinitialize swarm;
20.     end-if
21.     for each LSA i in lsas do
22.         localSearch(lsas[i]);
23.     end-for
24. until a termination condition is met
```

3.1.3. Detection and response to environmental changes

Several methods have been suggested for change detection. In this work, the best particle of the swarm is used as the “sentry” particle to detect the changes of the environment. In fact, the fitness value of the best particle is re-evaluated in each iteration. If the fitness of the best particle differs from its previous value, we can conclude that a change has occurred in the environment.

Once an environmental change is detected, all LSAs are re-evaluated and the swarm is randomly initialized in the search space. Moreover, the step size for LSAs is adjusted to its initial vale.

3.1.4. Global search, exploration and locating the promising areas of the search space

The particle swarm optimization used in this study is responsible for accomplishing a three-fold task: (a) exploring and detecting new promising areas of the search space, (b) placing a new LSA after converging to a previously undetected region, (c) controlling the density of the LSAs in different areas. In the following, the major components for accomplishing the mentioned tasks are described in detail.

3.1.4.1. Fuzzy social-only PSO model for detecting promising areas. Wang et al. [35] introduced a *fuzzy cognition local search* (FCLS) operator based on the *cognition-only model* of PSO to generate the velocity vector of the particles. Their experimental results show that FCLS can improve the performance of PSO in DOPs. Inspired by this work, in order to enhance the exploration ability of the swarm and avoiding the stagnation problem, a *fuzzy social-only model* is presented in this paper which is described as follows:

$$\vec{g}_i'(t) = N(\vec{g}(t), \sigma_i) \tag{8}$$

$$\vec{v}_i(t+1) = w\vec{v}_i(t) + c_2 r_2(\vec{g}_i'(t) - \vec{x}_i(t)) \tag{9}$$

where $\vec{g}_i'(t)$ is the fuzzy position of *gbest* for *i*th particle at time *t*, which is characterized by a normal distribution vector $N(\vec{g}_i(t), \sigma_i)$. σ_i is a parameter which controls the distribution range of vector *N* based on the normalized distance of the *i*th particle to the best particle which is defined as follows:

$$\sigma_i = 1 - \frac{d(\vec{x}_i, \vec{g})}{\sum_{popsize}^{j=1} d(\vec{x}_j, \vec{g})} \tag{10}$$

where $d(\vec{x}_i, \vec{g})$ denotes the Euclidean distance between particle *i* and the best particle of the swarm, and $\sum_{j=1}^{popsize} d(\vec{x}_j, \vec{g})$ is the total distance of all particles to the best particle of the swarm. The Euclidean distance between any two particles *i* and *j* in the *D*-dimensional search space is defined as follows:

$$d(i, j) = \sqrt{\sum_{d=1}^{D} (x_{id} - x_{jd})^2} \tag{11}$$

In this model, the nearer the LSA to the *gbest*, the larger the distribution range of vector *N*, and vice versa.

3.1.4.2. Check the status of the swarm. Various diversity measures exist in the literature for detecting the convergence of the swarm. For example, Blackwell [2] defined the *swarm diameter* as a criterion to determine the convergence of the swarm. The swarm diameter $|S|$ is defined as the largest distance, along any dimension, between any two particles of the swarm. If the swarm diameter is less than a threshold parameter, one can conclude that the swarm has converged. In the present study, we use the distance between any particle and $\vec{p}_g$ of the swarm to examine the convergence. The swarm has converged if the distance between every particle and $\vec{p}_g$ is less than a threshold R_{min}. The checking mechanism for convergence of the swarm is as shown in Algorithm 3.

Algorithm 3. *hasConverged*(swarm)

```
1. convergence := TRUE;
2. for each particle i in the swarm do
3.     dist := $\sqrt{\sum_{d=1}^{D}(x_{id} - p_{gd})^2}$, where $x_{id}$ is the value of the $d^{th}$ dimension of $i^{th}$ particle, and $p_{gd}$ is
       the value of the $d^{th}$ dimension of the best memory of the swarm;
4.     if (dist > $R_{min}$) then
5.         convergence := FALSE;
6.         break;
7.     end-if
8. end-for
9. return convergence;
```

3.1.4.3. Controlling the density of the LSAs. Several studies have confirmed that crowding of individuals in the same sub-area of the search space is not effective. In order to prevent LSAs from crowding in the search space and hence saving computing resources, a diversity control mechanism is introduced. In this regard, when the swarm converges to a position that has been already captured by a number of LSAs, it keeps the fittest LSA and removes the others from the search space. Algorithm 4 shows the density control mechanism in PSOLS.

Algorithm 4. *densityControl*(*lsas*)

```
1. Create a temporary list G of local search agents in the radius $\xi$ from $\vec{p}_g$;
2. Best_LSA := G[1];
3. for i := 2 to |G| do // finding the best LSA in the radius $\xi$ from $\vec{p}_g$
4.     if(G[i] is better than Best_LSA
5.         Best_LSA := G[i];
6.     end
7. end-for
8. Keep the Best_LSA and remove the other LSAs from lsas which are located in the radius $\xi$ from $\vec{p}_g$;
```

3.1.5. Local search, exploitation and tracking the optima

After locating a peak by the swarm, a local search agent is created by transferring the information of the best particle of the swarm to the LSA. The newly created LSA is then responsible for finding the top of the peak using a local search, and following the respective peak upon detecting a change in the environment. Any local search method can be used in this stage. In this study we apply different local search methods and investigate the effect of them on the performance of the proposed approach. These local search methods include a simple Random Walk (RW), Evolution Strategy (ES) [36], and Naive Directed Search (NDS) [26]. For detailed descriptions about RW, ES and NDS, interested readers are referred to the respective papers.

Several variations of PSOLS can be obtained by using different local search methods. In these variations when we refer to PSO-NDS, for example, we mean that we use NDS in the local search stage. For the sake of completeness and because of the critical role that NDS plays in our approach, here we give the pseudo-code for NDS in Algorithm 5.

Algorithm 5. Naive Directed Search Algorithm

```
1.  Initialize the position of the base point $X$ randomly
2.  $stepCounter := 0$;
3.  for each dimension $j$ do
4.      select a value for $Dir[j]$ from {-1,1} randomly
5.  end-for
6.  failedDims := $NULL$;
7.  repeat
8.      $stepSize := initialStepSize \cdot discountFactor^{stepCounter}$;
9.      for each Dimension $j$ that $j \notin failedDims$ DO
10.         $X_j' = X_j + Dir[j] \cdot stepSize$
11.         if fitness of $X'$ is greater than fitness of $X$ then
12.             $X := X'$;
13.         else
14.             $Dir[j] := -Dir[j]$;
15.             $X_j' = X_j + Dir[j] \cdot stepSize$;
16.             if fitness of $X'$ is greater than fitness of $X$ then
17.                 $X := X'$;
18.             else
19.                 Add $j$ to $failedDims$ Set;
20.             end-if
21.         end-if
22.     end-for
23.     if $|failedDims|$ is equal to the number of dimensions of the problem space then
24.         Increase $stepCounter$ by One;
25.         $failedDims := NULL$;
26.     end-if
27. until $stepSize >$ desiredStepSize
```

3.2. *Competitive PSOLS (CPSOLS)*

A dynamic optimization problem can contain several peaks or local optima. However, as the heights of the peaks are different, they do not have the same importance from the optimality point of view. In PSOLS, all available resources, i.e. function evaluations, are equally distributed between LSAs. It means that, LSAs consume an equal portion of the function evaluations regardless to the height of the peaks where they stand on.

As the first attempt to improve the performance of the proposed PSOLS, we try to allocate more function evaluations to the best performing LSA. The primary goal of the new adaptation is to let the best performing LSA perform extra local searches.

The advantage of this adaptation is that better performing LSAs will receive more function evaluations that would otherwise have been spent on exploiting the maximum of the sub-optimal peaks. In short, competitive PSOLS operates as follows:

(i) At the commencement of the run, allow the PSOLS to run in its standard manner.
(ii) Let the best LSA, i.e. the LSA with the highest fitness value, perform an extra local search and return to step (i).

The extra local search operation in the best LSA can be performed using any local search method and it should not be necessarily same as the local search in other LSAs. In this regard, various combinations of different local search methods are possible. Here, for example, CPSO(ES-NDS) means that the type of the local search in all LSAs is ES, and the extra local search on the best LSA is performed using NDS.

3.3. *Hibernating PSOLS (HPSOLS)*

The motivation behind this adaptation is to avoid allocating function evaluations to the LSAs that are not contributing much to the search process, hence allocating more fitness evaluations to the most successful LSAs. Therefore, HPSOLS differs from PSOLS in that local search operations are temporarily stopped in LSAs with step size less than S_{min}. The working procedure of HPSOLS can be summarized as follows:

(i) At the commencement of the run, allow the PSOLS to run in its standard manner.
(ii) If the step size of a LSA is less than the threshold S_{min}, then make it inactive.
(iii) If no environmental change has been detected, return to step (i).
(iv) Upon detecting a change in the environment, go to step (v).
(v) Activate all frozen LSAs, return to step (i).

It is worth noticing that the hibernation mechanism has been previously introduced by Kamosi et al. [28] at the swarm degree. Similar strategies were also used by the other authors for controlling the number of active swarms during the run [29,37]. The major difference between the proposed mechanism in this work and those introduced in mentioned papers is that in this work we apply the hibernation at individual degree, which is one of the contributions of this paper.

3.4. Competitive hibernating PSOLS (CHPSOLS)

The basis for the new adaptation is to combine desirable features of both HPSOLS and CPSOLS together. Since HPSOLS and CPSOLS are independent adaptations, we can use both of them simultaneously. The CHPSOLS can be shortly described as follows:

(i) At the commencement of the run, allow the PSOLS to run in its standard manner.
(ii) If the step size of a LSA is less than the threshold S_{min}, then make it inactive.
(iii) Upon detecting a change in the environment, go to step (iv).
(iv) Activate all frozen LSAs, return to step (v).
(v) Let the best LSA to perform an extra local search and return to step (i).

4. Experimental study

In this section, three groups of experiments are carried out in order to evaluate the performance of our proposed PSOLS. In the first group of experiments, we investigate the influence of different components and parameters of PSOLS on the performance. The second group of experiments is devoted to an in depth comparison of PSOLS with some PSO-based algorithms in numerous dynamic scenarios modeled by MPB. Finally, in the third group of experiments, we examine the performance of our proposed PSOLS in comparison with several recent and well-known methods in the literature on some of the most commonly used configurations of MPB.

4.1. Experimental setup

4.1.1. Dynamic test function

One of the most widely used synthetic dynamic optimization test suites in the literature is the moving peaks benchmark (MPB) proposed by Branke [38], which is highly regarded due to its configurability. MPB is a real-valued dynamic environment with a *D*-dimensional landscape consisting of *m* peaks, where the height, the width and the position of each peak are changed slightly every time a change occurs in the environment [39]. Different landscapes can be defined by specifying the shape of the peaks. A typical peak shape is conical which is defined as follows:

$$f(\vec{x}, t) = \max_{i=1,\ldots,m} H_t(i) - W_t(i)\sqrt{\sum_{j=1}^{D}(x_t(j) - X_t(i,j))^2} \quad (12)$$

where $H_t(i)$ and $W_t(i)$ are the height and the width of peak *i* at time *t*, respectively. The coordinates of each dimension $j \in [1, D]$ related to the location of peak *i* at time *t*, are expressed by $X_t(i, j)$, and *D* is the problem dimensionality. A typical change of a single peak can be modeled as follows:

$$H_{t+1}(i) = H_t(i) + height_{severity} \cdot \sigma_h \quad (13)$$

$$W_{t+1}(i) = W_t(i) + width_{severity} \cdot \sigma_w \quad (14)$$

$$\vec{X}_{t+1}(i) = \vec{X}_t(i) + \vec{v}_{t+1}(i) \quad (15)$$

$$\vec{v}_{t+1}(i) = \frac{s}{|\vec{r} + \vec{v}_t(i)|}((1-\lambda)\vec{r} + \lambda\vec{v}_t(i)) \quad (16)$$

where σ_h and σ_w are two random Gaussian numbers with zero mean and standard deviation one. Moreover, the shift vector $\vec{v}_{t+1}(i)$ is a combination of a random vector $\vec{r}$, which is created by drawing random numbers in $[-0.5, 0.5]$ for each dimension, and the current shift vector $\vec{v}_t(i)$, and normalized to the length *s*. Parameter $\lambda \in [0.0, 1.0]$ specifies the correlation of each peak's changes to the previous one. This parameter determines the trajectory of changes, where $\lambda = 0$ means that the peaks are shifted in completely random directions and $\lambda = 1$ means that the peaks always follow the same direction, until they hit the boundaries where they bounce off.

Different instances of the MPB can be obtained by changing the environmental parameters. Three sets of configurations have been introduced to provide a unified test bed for the researchers to investigate the performance of their approaches in the same condition. Among them, the second configuration (Scenario 2) is the most widely used configuration, which was also used as the base configuration for the experiments conducted in this paper.

4.1.2. Performance measures

For the purpose of measuring the efficiency of the optimization algorithms on DOPs, we use *offline error* which is the most well-known metric for dynamic environments. There are two measures in the literature which have been termed as "offline error". The first one is the performance measure suggested by Branke and Schmeck [40] which is defined as the average of the smallest error found since the last change in the environment over the entire run as follows:

$$E_O = \frac{1}{T}\sum_{t=1}^{T} e_t^* \quad (17)$$

where *T* is the maximum number of function evaluations so far and e_t^* is the minimum error gained by the optimization algorithm since the last change at the *t*th fitness evaluation.

The other measure, which was first proposed in [41] as *accuracy* and later named as *best error before change* by Nguyen et al. [42], is calculated as the average of the minimum fitness error achieved by the algorithm at the end of each period right before the moment of change, as follows [21]:

$$E_B = \frac{1}{K}\sum_{k=1}^{K}(h_k - f_k) \quad (18)$$

where f_k is the best solution obtained by the algorithm just before the *K*th change occurs, h_k is the optimum value of the *K*th environment and *K* is the total number of environments. In this study, we use both measures to evaluate the performance of the proposed approach. For more detail on the difference between these two measures, interested readers are referred to [43].

4.1.3. Experimental settings

For PSOLS, a swarm with three particles is applied. The learning factors c_2 is set to 1.496180 and inertia weight *w* is set to 0.729844. The convergence radius R_{min} and radius ξ are set to 10 and 20, respectively. For NDS local search, the initial step size δ_{init}, discount factor *b*, and minimum step size S_{min} are set to 0.5, 0.2, and 0.01. When using ES and RW, the initial step size σ_{init} is 0.2. For ES, $(1+1)$ES with 1/5 success rule is applied.

Unless stated otherwise, the default value for the parameters of the proposed approach are according to Table 2. For each experiment of the proposed algorithm on a specific DOP, at least 500 independent runs were executed with different random seeds. For each run of the algorithm, 100 environmental changes were considered as the termination condition, which results in $f \times 100$ function evaluations. The experimental results are reported in terms of average offline error (best error before change) and standard error, which is calculated as standard deviation divided by the squared root of the number of runs. Moreover, in all experiments of this paper the results of the best performing algorithm, which are significantly superior to the others with a *t*-test at 5% significant level, are printed in boldface. In order to counteract the effect of type I error, Holm–Bonferroni correction [44] has been applied to control the family-wise error rate. In situations where the outcomes of the

Table 1
Parameter settings for the Moving Peaks problem (Scenario 2).

Parameter	Default value	Other tested values
Number of peaks (m)	10	1, 5, 20, 30, 40, 50, 100, 200
Height severity	7.0	
Width severity	1.0	
Peak function	Cone	
Number of dimensions (d)	5	10, 20, 50
Height range (H)	∈[30, 70]	
Width range (W)	∈[1, 12]	
Standard height (I)	50.0	
Search space range (A)	$[0, 100]^d$	
Frequency of change (f)	5000	500, 1000, 2500, 10,000
Shift severity (s)	1	0, 2, 3, 4, 5
Correlation coefficient (λ)	0.0	
Basic function	No	

t-test reveal that the results of the best performing algorithms are not statically significant, all of them are printed in boldface.

4.2. Experimental investigation of PSOLS

4.2.1. Parameter sensitivity analysis for PSOLS in dynamic environments

In this sub-section, effects of key parameters (i.e. swarm size, R_{min}) and components (i.e. local search operation, function evaluation management scheme) on the performance of PSOLS were analyzed on different DOPs.

4.2.1.1. Effect of the swarm size. In this set of experiments, we examine the effect of the swarm size on the performance of PSOLS in different dynamic environments. Various experiments were carried out with value of the swarm size chosen from [3,14]. The environmental parameters were the combination of change frequency $f \in \{500, 1000, 2500, 5000\}$ and the number of peaks $m \in \{1, 3, 5, 10, 20, 100\}$.

Fig. 1 draws the average offline error of PSOLS with varying swarm size across different dynamic environments. From Fig. 1, it can be seen that the swarm size 3 produces a better result in most of the tested DOPs. It can be also seen that the performance of PSOLS is degraded as the swarm size increases. The reason mainly lies in the fact that the swarm is responsible for creating LSAs in promising areas of the search space. The bigger the population size of the swarm, the more function evaluations is consumed by the swarm for converging to an area. This makes less function evaluations available for LSAs to exploit the peaks.

Our experiments revealed that, the standard PSO with only three particles fails to achieve a good performance due to the stagnation. However, the proposed fuzzy social-only model can improve the exploration ability of PSO by its random behavior.

4.2.1.2. Effect of the radius R_{min}. The aim of this experiment is to investigate the effect of R_{min} on the performance of PSOLS. As we mentioned in Section 3, R_{min} is a threshold parameter which is used to determine the convergence of the swarm. Therefore, it is expected that this parameter has a significant impact on the performance of PSOLS.

From Table 3, it can be seen that PSOLS with $R_{min} = 10.0$ gives better results. On the one hand, a large value of R_{min} causes that the swarm creates LSAs in areas far away from the peaks. On the other hand, a very small value of R_{min} causes that the swarm consumes a large number of fitness evaluations to converge. In addition, a small value of R_{min} can cause the swarm to frequently converge to a limited area of the search space. Table 3 also indicates that the extreme values for this parameter, i.e. 0.01, 0.1 and 100, gives the worst results.

Table 2
Default parameter settings for PSOLS.

Parameter	Value
Number of particles in the swarm	3
c_2	1.496180
w	0.729844
R_{min}	10
ξ	20.0
δ_{init}	0.5
b	0.2
S_{min}	0.01
σ_{init}	0.2

4.2.2. Effect of using different local search methods

This set of experiments aims to investigate the effect of local search method being used by each LSA on the performance of PSOLS in DOPs with different complexities. The environmental conditions were different combinations of dimension $d \in \{5, 10, 100\}$, number of peak $m \in \{1, 10, 100\}$, and severity $s \in \{0.0, 1.0, 2.0, 3.0, 4.0, 5.0\}$. The results are given in Table 4. From Table 4, it is observed that NDS is the most favorable local search method in the majority of the tested DOPs.

4.2.3. Effect of collaboration between swarm and local search agents

This experiment has been specifically designed to verify the effectiveness of hybridizing PSO with local search agents. In this experiment, the performance of the proposed method is compared with two different methods: (1) re-randomization PSO, i.e. RPSO and (2) a group of independent local search agents (GILSAs). For RPSO, a population size of 30 particles was used. Furthermore, the velocity of particles was clamped within the range [−20, 20] and the rate of re-randomization was set to 50%. For GILSAs, a number of LSAs equal to the number of peaks in the landscape are randomly initialized into the search space. Each LSA then starts to explore the search space using NDS. To prevent LSAs from searching in the same area, the exclusion operator with radius 20 is also applied.

The results of experiment conducted to investigate the effectiveness of hybridizing PSO with local search agents are listed in Table 5.

As shown in Table 5, all the results of the proposed hybrid method are much better than the results of RPSO and GILSAs. The comparison results clearly confirm the advantages of collaboration between the swarm and LSAs in PSOLS.

4.2.4. Effect of different resource management schemes

The most important component of the proposed adapted algorithms is the resource management scheme that increases the efficiency of the PSOLS. Therefore, experiments were carried out to validate the benefits of the proposed adaptations. The main goal of this set of experiments is to specifically emphasize on the importance of the managing function evaluations for DOPs. Furthermore, we are interested in investigating the effect of combining different resource management schemes. The performance of different algorithms was evaluated on various DOPs with $f \in \{500, 2500, 5000\}$, $d \in \{5, 10\}$, and $m \in \{1, 10, 100\}$. Experimental results are provided in Table 6. Table 6 indicates that managing function evaluations is a major concern that can contribute much to the performance of PSOLS on dynamic environments.

Several conclusions can be drawn from Table 6. Here we highlight a number of important conclusions as follows:

(1) It is clearly observed that the resource management schemes have a significant influence on the performance of the proposed approach.

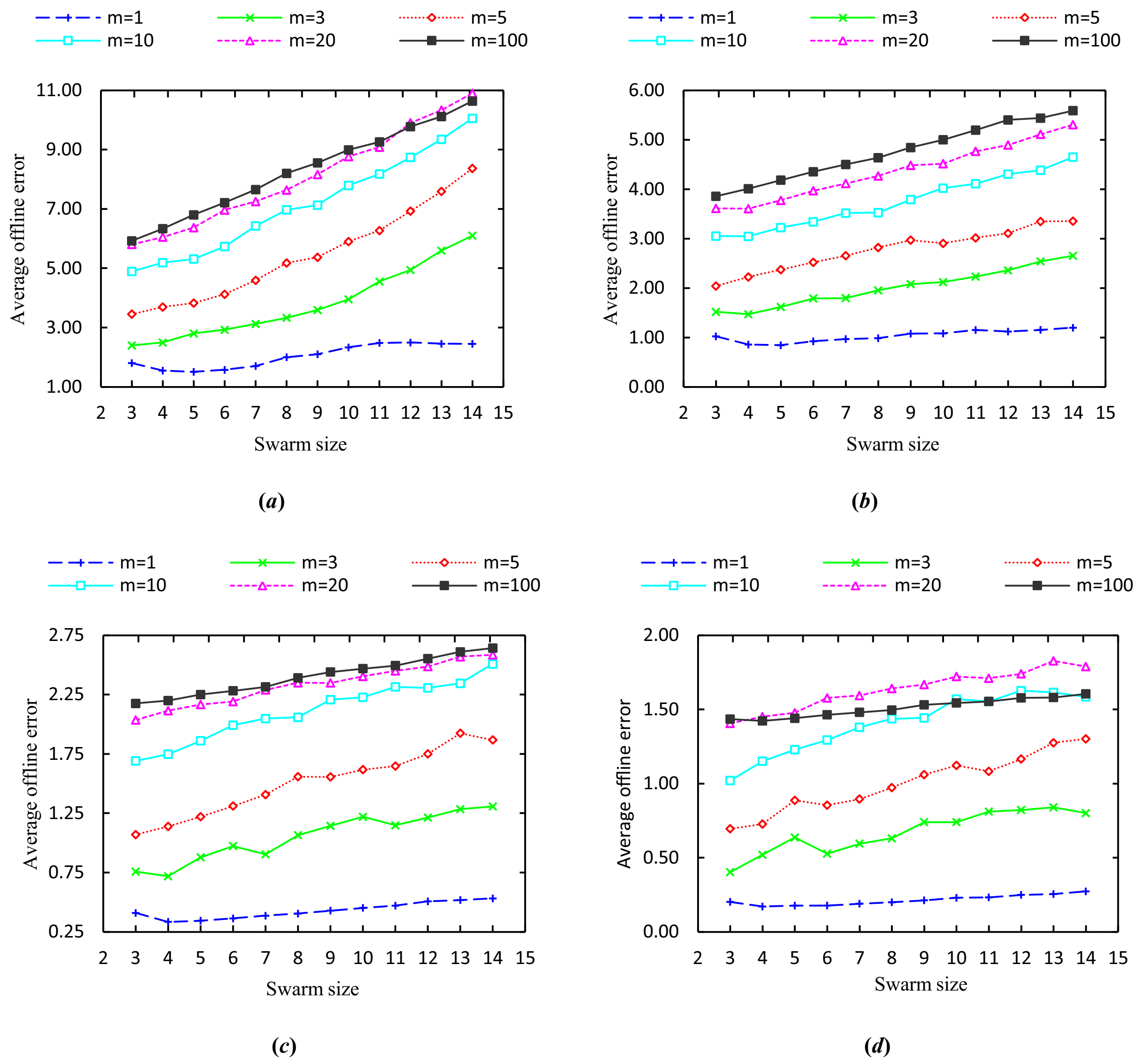

Fig. 1. Offline error of PSOLS with different swarm size on dynamic environments with change frequency: (a) $f = 500$, (b) $f = 1000$, (c) $f = 2500$, and (d) $f = 5000$.

(2) The performance of CPSOLS(NDS-NDS) is worse than PSO-NDS. The reason is that NDS is a very expensive local search that consumes many function evaluations.

(3) Combining different local search methods in CPSOLS is preferable. Different local search methods have different characteristics. ES local search can improve the position of the LSAs with consuming fewer resources. Afterwards, NDS can exploit around the best found area more elaborately.

(4) The proposed hibernation mechanism can improve the performance of the proposed algorithm by stopping the local search in LSAs that are not beneficial for the search process.

4.3. Comparison with other state-of-the-art PSO variants

In this sub-section, various experiments were carried out to compare the performance of the proposed method with several PSO-based algorithms, on different DOPs generated by MPB. The

Table 3
Effect of threshold R_{min} on the offline error of PSOLS in DOPs with varying number of peaks and $f = 5000$.

R_{min}	*m*								
	1	5	10	20	30	40	50	100	200
0.01	0.170449	1.472404	2.288167	2.754353	2.943040	3.051148	3.128575	3.193482	3.071423
0.1	0.163543	1.206962	1.865395	2.276448	2.422306	2.530776	2.578950	2.681032	2.582787
1	**0.144271**	0.936765	1.529951	1.863245	1.904520	1.907127	1.966856	1.964764	1.957201
10	0.175372	**0.699090**	1.065045	**1.388958**	**1.462347**	**1.462515**	**1.461377**	**1.419923**	**1.396562**
20	0.332701	0.726745	**1.005591**	1.394626	1.532482	1.521356	1.528934	1.553289	1.546568
30	0.563210	0.902228	1.184510	1.529234	1.709050	1.762999	1.810098	1.845009	1.814408
40	0.711218	0.988773	1.247041	1.668323	1.822326	1.937583	1.988288	2.042053	1.966935
50	0.864800	1.085577	1.354654	1.827569	1.985264	2.076446	2.137183	2.158379	2.102635
100	1.445447	1.574296	1.694178	2.197759	2.454605	2.558030	2.648966	2.644556	2.518883

Table 4
Effect of using different local search methods on the offline error of PSOLS.

Method	d	m	Shift length (s)					
			0.0	1.0	2.0	3.0	4.0	5.0
PSO-RW	5	1	0.45112	0.83071	1.40234	2.30422	3.58285	5.12686
		10	0.86069	1.58847	2.76093	4.22373	6.14465	8.35998
		100	0.78117	2.24661	4.18529	6.35894	8.65815	11.0524
	10	1	6.39022	8.76646	11.1169	15.4064	22.0227	28.5702
		10	6.55826	9.86580	14.9211	20.9969	27.3431	34.5071
		100	4.76478	8.33540	13.0154	18.6546	24.6568	29.7344
	50	1	850.497	836.136	862.561	846.742	870.700	875.359
		10	341.455	337.526	338.978	338.985	341.808	341.972
		100	216.296	216.764	216.905	216.806	217.633	217.614
PSO-ES	5	1	0.18182	0.41204	0.65256	0.87743	1.17967	1.43267
		10	**0.61517**	1.10621	1.56895	1.96668	2.30917	2.69568
		100	**0.56870**	1.72911	2.52528	3.14339	3.64598	4.04492
	10	1	**0.50642**	1.15019	1.84503	2.44935	3.07189	3.73131
		10	**1.99820**	**3.67536**	5.02192	6.19992	7.26207	8.29264
		100	**1.14821**	**3.42274**	5.00106	6.34553	7.62520	8.77980
	50	1	8.31006	16.2051	23.5525	31.0304	38.1722	47.5813
		10	**8.92927**	**14.9613**	20.1166	24.0602	28.0973	31.5220
		100	**5.74614**	**10.9234**	14.4799	17.6129	20.4908	23.0889
PSO-NDS	5	1	**0.16546**	**0.20380**	**0.24179**	**0.29319**	**0.36372**	**0.44270**
		10	0.85333	**1.03259**	**1.16308**	**1.36187**	**1.52842**	**1.79575**
		100	1.11648	**1.45459**	**1.79764**	**2.18646**	**2.57144**	**2.98087**
	10	1	0.57711	**0.72697**	**0.89392**	**1.01887**	**1.22884**	**1.50074**
		10	4.26594	4.11061	**4.61842**	**5.03542**	**5.65114**	**6.39365**
		100	3.38095	3.37294	**3.70708**	**4.25223**	**4.90236**	**5.43897**
	50	1	**7.27152**	**8.16682**	**8.88441**	**9.15409**	**9.75214**	**10.5055**
		10	15.3540	16.3572	**16.9831**	**17.3668**	**17.2677**	**18.1880**
		100	12.4533	13.8362	**14.1176**	**14.4144**	**15.1625**	**15.5804**

Table 5
Average offline error of RPSO, GILSAs, and PSOLS in DOPs with varying number of peaks and change intervals.

Method	f	Number of peaks (m)								
		1	5	10	20	30	40	50	100	200
RPSO	500	9.41	19.74	20.48	20.21	19.24	18.48	17.79	15.96	14.27
GILSAs		3.50	11.49	11.01	11.74	13.90	17.46	24.08	27.68	31.27
PSO-NDS		**2.31**	**4.86**	**5.87**	**6.23**	**6.28**	**6.14**	**6.12**	**6.02**	**5.74**
RPSO	1000	4.91	16.88	17.79	17.69	16.74	16.22	15.88	13.46	12.15
GILSAs		1.77	10.20	9.43	7.75	7.40	7.99	8.43	16.02	19.29
PSO-NDS		**1.13**	**2.61**	**3.61**	**3.98**	**4.01**	**3.94**	**3.95**	**3.89**	**3.71**
RPSO	2500	2.91	14.51	16.15	15.77	15.08	14.22	13.94	12.24	10.46
GILSAs		0.72	9.56	8.98	6.76	5.45	4.91	4.56	4.76	6.74
PSO-NDS		**0.45**	**1.12**	**1.90**	**2.18**	**2.29**	**2.25**	**2.24**	**2.21**	**2.13**
RPSO	5000	1.75	13.64	15.22	15.36	14.18	13.50	13.23	10.86	9.77
GILSAs		0.35	9.06	9.07	6.85	5.34	4.25	3.83	3.05	3.34
PSO-NDS		**0.22**	**0.57**	**1.13**	**1.50**	**1.53**	**1.53**	**1.50**	**1.44**	**1.40**

involved algorithms include mQSO [18], AmQSO [45], CellularPSO [24], mPSO [27], HmSO [28], APSO [46], CPSO [21] and CPSOR[1] [16], TP-CPSO [26], and FTMPSO [29]. Among the peer algorithms, the mQSO and AmQSO were implemented for this study as described in the corresponding papers. For CPSO and CPSOR on each DOP, the best configuration of the algorithms which enables them to achieve their best performance is adopted. The results of TP-CPSO were also updated for the purposes of this paper. For mQSO and AmQSO, the parameter settings that were suggested in the original paper were also applied here.

4.3.1. *Effect of varying the number of peaks and change frequencies*

The experiment aims to investigate the performance comparison between CHPSO(ES-NDS) and 10 PSO-based algorithms on DOPs with different number of peaks $m \in \{1, 5, 10, 20, 30, 40, 50, 100, 200\}$ and change frequencies $f \in \{500, 1000, 2500, 5000, 10{,}000\}$. The other environmental parameters are set according to the MPB settings shown in Table 1. The various combinations of (f, m) resulted in 45 different DOPs being modeled by the MPB. The numerical results of different PSO-based algorithms are reported in Tables 7 and 8.

From Tables 5 and 6, it can be clearly observed that our proposed method is superior to the other contestant algorithms on most of the tested DOPs. The results confirmed that our approach can successfully combine the explorative ability of the population-based method and exploitative capability of the local search in a single method.

4.3.2. *Effect of varying the shift length severity*

In this experiment, the effect of change severity on the performance of different algorithms is examined.

[1] Source code for CPSO and CPSOR is publicly available at: http://cs.cug.edu.cn/teacherweb/lichanghe/pages/EAlib.html.

Table 6
Offline error of PSOLS with different resource management schemes on DOPs with different environmental dynamics.

Method	PSO-ES	PSO-NDS	CPSO(NDS-NDS)	CPSO(ES-NDS)	CHPSO(NDS-NDS)	CHPSO(ES-NDS)
d, m	*f* = 500					
5, 1	3.97	1.77	1.78	**1.56**	1.93	1.58
5, 10	5.25	5.16	5.47	3.56	3.58	**3.54**
5, 100	5.38	6.02	6.22	3.79	4.10	**3.78**
10, 1	9.29	5.65	6.84	**4.76**	8.00	4.80
10, 10	10.84	13.23	14.64	**8.46**	9.77	8.55
10, 100	9.13	12.04	13.80	**7.45**	8.40	7.55
Average	7.31	7.31	8.12	**4.93**	5.96	4.97
d, m	*f* = 2500					
5, 1	0.83	0.41	0.40	**0.36**	0.47	0.38
5, 10	1.90	1.72	1.85	1.31	1.21	**1.23**
5, 100	2.50	2.19	2.22	1.53	1.49	**1.50**
10, 1	2.38	1.42	1.67	0.99	1.88	**0.98**
10, 10	5.04	5.89	6.40	**3.35**	3.81	**3.35**
10, 100	4.50	4.92	5.08	**2.71**	3.25	**2.71**
Average	2.86	2.76	2.94	1.71	2.02	**1.69**
d, m	*f* = 5000					
5, 1	0.41	0.19	0.20	**0.18**	0.23	0.19
5, 10	1.11	1.07	1.12	0.83	0.75	**0.75**
5, 100	1.73	1.46	1.47	1.05	0.99	**1.03**
10, 1	1.15	0.72	0.82	**0.48**	0.93	0.51
10, 10	3.68	4.20	4.24	2.51	3.02	**2.43**
10, 100	3.42	3.33	3.39	**1.99**	2.82	2.00
Average	1.92	1.83	1.87	1.17	1.46	**1.15**

As shown in Table 9, the performance of CHPSO(ES-NDS) is superior to the other methods in five out of six DOPs. In a DOP with $s = 5$, the offline error achieved by FTMPSO is slightly lower than that achieved by CHPSO(ES-NDS). The reason is in the fact that FTMPSO utilizes the information about the shift severity, i.e. s, to respond to environmental changes, whereas this information may not be available in real-world situations.

As can be seen in Table 10, the best error before change in CHPSO(ES-NDS) is less affected by the change severity of the environment.

4.3.3. *Effect of varying the number of dimensions*

Many real-world problems consist of optimizing a large number of components. In this experiment, the performance of the eight PSO algorithms is investigated on the MPB with different dimensionalities $d \in \{5, 10, 20, 50\}$ and other dynamic and complexity parameters are set according to Table 1.

Regarding the results in Table 11, CHPSO(ES-NDS) can significantly outperform the other PSO variants. Results also indicate that our method is less affected by the number of dimensions.

Comparing the results of CHPSO(ES-NDS) with CPSO and CPSOR in Table 12, the best error before change achieved by CHPSO(ES-NDS) is much less than that achieved by CPSO in all tested environments. Although on DOPs with $d = 5$, and $d = 10$ the results of CHPSO(ES-NDS) is slightly worse than those obtained by CPSOR, on DOPs with a large number of dimensions they are much better than the results of CPSOR. It should be noted that CPSO and CPSOR were reconfigured for each DOP separately, but the same settings were used for CHPSO(ES-NDS) in all experiments.

4.3.4. *Convergence behavior of the proposed approach*

This experiment aims to provide a visual view of the convergence behavior of CHPSO(ES-NDS). Fig. 2 illustrates a graphical representation of the convergence of different PSO-based algorithms in a DOP with 50 peaks and change frequency 5000.

Regarding Fig. 2, the first thing that stands out is that AmQSO and CellularPSO have the poorest performance during the whole search process. From the curves it is noticeable that the small population of CHPSO(ES-NDS) can be favorable to its current error at the very early stages of the optimization. Later, from evaluation 2400 to 30,000, TCPSO can perform slightly better than CHPSO(ES-NDS). The reason is that TCPSO can locate the peaks of the landscape faster. However, once the peaks are discovered, the powerful exploitation ability of the CHPSO(ES-NDS) can be very desirable in both refining the solutions and following the trajectories of the peaks after the change.

4.4. *Comparison with other dynamic optimization methods*

In this sub-section, we study the performance of the proposed method in comparison with several recently proposed algorithms taken from the literature. The involved algorithms include:

- The differential evolution with dynamic population adjustment for unknown environments (DynPopDE) [47].
- The competing differential evolution (CDE) [48].
- A modified multi-population artificial fish swarm algorithm (DMAFSA) [49].
- A bi-population hybrid approach (CDEPSO) [34].
- A memetic particle swarm optimizer (MPSO) [50].
- A competitive clustering particle swarm optimizer (CCPSO) [22].
- Two multi-population evolutionary algorithms with clustering scheme (CGAR and CDER) [16].
- The finder-tracker multi-swarm PSO (FTMPSO) [29].
- The speciation based firefly algorithm (SFA) [51].
- An improved artificial fish swarm algorithm (mNAFSA) [52].
- A multi-population electromagnetic algorithm (EM-POP) [53].

The results of the peer algorithms shown in Tables 13 and 14 are borrowed from their original reference. For each algorithm on a specific environment, the standard deviation, or confidence interval, has been converted to the standard error.

From Table 13, it can be realized that the proposed CHPSO(ES-NDS) is considerably better than the other peer algorithms in most of the tested DOPs in terms of offline error.

Table 7
Average offline error ± standard error of nine PSO-based algorithms on DOPs with $d = 5$, $s = 1$ and varying number of peaks and change intervals.

f	m	Method								
		mQSO	AmQSO	CellularPSO	mPSO	HmSO	APSO	TP-CPSO	FTMPSO	CHPSO(ES-NDS)
500	1	40.30 ± 1.30	10.41 ± 0.34	22.37 ± 3.87	8.71 ± 0.48	9.40 ± 0.31	4.81 ± 0.14	5.96 ± 0.22	1.76 ± 0.09	**1.68 ± 0.03**
	5	11.50 ± 0.21	6.95 ± 0.12	14.20 ± 0.65	6.69 ± 0.26	7.83 ± 0.17	4.95 ± 0.11	5.12 ± 0.11	**2.93 ± 0.18**	2.97 ± 0.06
	10	8.96 ± 0.19	6.97 ± 0.13	13.55 ± 0.52	7.19 ± 0.23	8.04 ± 0.14	5.16 ± 0.11	5.56 ± 0.11	3.91 ± 0.19	**3.74 ± 0.05**
	20	8.29 ± 0.11	7.61 ± 0.10	12.77 ± 0.38	8.01 ± 0.19	8.25 ± 0.12	5.81 ± 0.08	5.67 ± 0.10	4.83 ± 0.19	**3.93 ± 0.04**
	30	8.32 ± 0.09	8.99 ± 0.14	12.55 ± 0.42	8.43 ± 0.17	8.62 ± 0.12	6.03 ± 0.07	5.59 ± 0.08	5.05 ± 0.21	**3.98 ± 0.03**
	40	8.18 ± 0.08	10.26 ± 0.15	12.33 ± 0.38	8.62 ± 0.18	8.67 ± 0.12	6.10 ± 0.08	5.53 ± 0.07	–	**3.92 ± 0.03**
	50	8.12 ± 0.08	11.45 ± 0.17	12.19 ± 0.31	8.76 ± 0.18	8.81 ± 0.12	5.95 ± 0.06	5.57 ± 0.08	4.98 ± 0.15	**3.86 ± 0.03**
	100	7.81 ± 0.08	12.35 ± 0.15	11.38 ± 0.29	8.91 ± 0.17	8.88 ± 0.11	6.08 ± 0.06	5.36 ± 0.06	5.31 ± 0.11	**3.81 ± 0.02**
	200	7.59 ± 0.07	12.82 ± 0.13	11.34 ± 0.27	8.88 ± 0.14	9.06 ± 0.11	6.20 ± 0.04	5.33 ± 0.05	5.52 ± 0.21	**3.68 ± 0.02**
1000	1	21.42 ± 0.69	6.15 ± 0.20	7.97 ± 0.54	4.44 ± 0.24	4.89 ± 0.16	2.72 ± 0.04	2.66 ± 0.08	**0.89 ± 0.05**	**0.90 ± 0.02**
	5	6.32 ± 0.13	4.40 ± 0.11	6.16 ± 0.31	3.93 ± 0.16	4.46 ± 0.08	2.99 ± 0.09	2.67 ± 0.09	**1.70 ± 0.10**	**1.71 ± 0.04**
	10	5.20 ± 0.10	4.50 ± 0.09	5.94 ± 0.23	4.57 ± 0.18	4.79 ± 0.07	3.87 ± 0.08	3.17 ± 0.06	**2.36 ± 0.09**	**2.35 ± 0.04**
	20	5.40 ± 0.08	4.74 ± 0.06	6.13 ± 0.17	4.97 ± 0.13	4.99 ± 0.08	4.13 ± 0.06	3.33 ± 0.06	3.01 ± 0.12	**2.62 ± 0.03**
	30	5.50 ± 0.06	4.79 ± 0.04	6.23 ± 0.15	5.15 ± 0.12	5.11 ± 0.07	4.12 ± 0.04	3.45 ± 0.06	3.06 ± 0.10	**2.63 ± 0.02**
	40	5.48 ± 0.07	4.80 ± 0.04	6.27 ± 0.15	5.17 ± 0.10	5.13 ± 0.06	4.15 ± 0.04	3.58 ± 0.05	–	**2.54 ± 0.02**
	50	5.31 ± 0.06	4.86 ± 0.04	6.26 ± 0.16	5.33 ± 0.10	5.15 ± 0.04	4.11 ± 0.03	3.57 ± 0.05	3.29 ± 0.10	**2.52 ± 0.02**
	100	5.09 ± 0.05	4.91 ± 0.04	6.27 ± 0.14	5.60 ± 0.09	5.28 ± 0.05	4.26 ± 0.04	3.52 ± 0.04	3.63 ± 0.09	**2.47 ± 0.01**
	200	4.84 ± 0.05	5.18 ± 0.04	6.01 ± 0.11	5.78 ± 0.09	5.43 ± 0.05	4.21 ± 0.02	3.51 ± 0.04	3.74 ± 0.09	**2.39 ± 0.01**
2500	1	9.48 ± 0.31	2.44 ± 0.07	4.57 ± 0.31	1.79 ± 0.10	2.02 ± 0.07	1.06 ± 0.03	0.92 ± 0.03	**0.39 ± 0.02**	**0.40 ± 0.01**
	5	3.33 ± 0.08	2.44 ± 0.08	3.15 ± 0.21	2.04 ± 0.12	2.12 ± 0.09	1.55 ± 0.05	1.17 ± 0.05	0.91 ± 0.08	**0.78 ± 0.03**
	10	2.85 ± 0.07	2.65 ± 0.05	3.09 ± 0.16	2.66 ± 0.16	2.43 ± 0.07	2.17 ± 0.07	1.59 ± 0.06	1.21 ± 0.06	**1.16 ± 0.02**
	20	3.41 ± 0.06	2.99 ± 0.05	3.60 ± 0.13	3.07 ± 0.11	2.63 ± 0.05	2.51 ± 0.05	1.82 ± 0.04	1.66 ± 0.05	**1.47 ± 0.02**
	30	3.45 ± 0.05	3.01 ± 0.03	3.88 ± 0.12	3.15 ± 0.08	2.73 ± 0.04	2.61 ± 0.02	1.99 ± 0.04	1.87 ± 0.05	**1.58 ± 0.02**
	40	3.45 ± 0.04	3.04 ± 0.03	4.17 ± 0.12	3.17 ± 0.07	2.77 ± 0.04	2.59 ± 0.03	1.96 ± 0.03	–	**1.56 ± 0.01**
	50	3.38 ± 0.04	3.03 ± 0.03	4.25 ± 0.12	3.26 ± 0.07	2.76 ± 0.04	2.66 ± 0.02	2.01 ± 0.03	2.09 ± 0.07	**1.58 ± 0.01**
	100	3.07 ± 0.03	2.96 ± 0.02	4.25 ± 0.13	3.31 ± 0.05	2.83 ± 0.03	2.62 ± 0.02	2.09 ± 0.03	2.22 ± 0.06	**1.49 ± 0.01**
	200	2.92 ± 0.02	2.93 ± 0.02	4.20 ± 0.09	3.36 ± 0.05	2.94 ± 0.02	2.64 ± 0.01	2.06 ± 0.02	2.22 ± 0.07	**1.44 ± 0.01**
5000	1	4.49 ± 0.14	1.27 ± 0.04	2.79 ± 0.19	0.90 ± 0.05	1.01 ± 0.03	0.53 ± 0.01	0.40 ± 0.01	**0.18 ± 0.01**	0.19 ± 0.00
	5	1.72 ± 0.05	1.30 ± 0.06	1.94 ± 0.18	1.21 ± 0.12	1.23 ± 0.05	1.05 ± 0.06	0.75 ± 0.07	0.47 ± 0.05	**0.44 ± 0.02**
	10	1.77 ± 0.05	1.64 ± 0.05	1.93 ± 0.08	1.66 ± 0.08	1.47 ± 0.04	1.31 ± 0.03	0.96 ± 0.05	0.67 ± 0.04	**0.64 ± 0.02**
	20	2.49 ± 0.05	2.03 ± 0.04	2.73 ± 0.12	2.05 ± 0.08	1.67 ± 0.04	1.69 ± 0.05	1.18 ± 0.03	0.93 ± 0.04	**0.91 ± 0.01**
	30	2.58 ± 0.04	2.11 ± 0.03	3.08 ± 0.11	2.18 ± 0.06	1.72 ± 0.03	1.78 ± 0.02	1.32 ± 0.03	1.14 ± 0.04	**0.99 ± 0.01**
	40	2.58 ± 0.04	2.18 ± 0.02	3.28 ± 0.11	2.24 ± 0.06	1.78 ± 0.03	1.86 ± 0.02	1.36 ± 0.03	–	**1.02 ± 0.01**
	50	2.50 ± 0.03	2.18 ± 0.02	3.34 ± 0.07	2.30 ± 0.04	1.75 ± 0.02	1.95 ± 0.02	1.40 ± 0.03	1.32 ± 0.04	**1.03 ± 0.01**
	100	2.27 ± 0.02	2.14 ± 0.01	3.48 ± 0.11	2.32 ± 0.04	1.78 ± 0.02	1.95 ± 0.01	1.50 ± 0.02	1.61 ± 0.03	**1.04 ± 0.01**
	200	2.12 ± 0.02	2.12 ± 0.01	3.37 ± 0.08	2.34 ± 0.03	1.82 ± 0.01	1.90 ± 0.01	1.56 ± 0.02	1.67 ± 0.03	**1.01 ± 0.00**
10,000	1	2.34 ± 0.07	0.60 ± 0.01	1.63 ± 0.12	0.44 ± 0.02	0.52 ± 0.02	0.25 ± 0.01	0.20 ± 0.01	**0.09 ± 0.00**	**0.09 ± 0.00**
	5	0.97 ± 0.04	0.75 ± 0.05	1.20 ± 0.15	0.72 ± 0.08	0.76 ± 0.07	0.57 ± 0.03	0.37 ± 0.03	0.31 ± 0.04	**0.25 ± 0.01**
	10	1.10 ± 0.03	1.01 ± 0.04	1.19 ± 0.09	1.05 ± 0.10	0.90 ± 0.04	0.82 ± 0.02	0.54 ± 0.03	0.43 ± 0.03	**0.40 ± 0.01**
	20	1.89 ± 0.05	1.35 ± 0.03	2.13 ± 0.10	1.37 ± 0.06	1.09 ± 0.03	1.23 ± 0.02	0.74 ± 0.02	0.56 ± 0.01	**0.52 ± 0.01**
	30	1.98 ± 0.03	1.44 ± 0.02	2.59 ± 0.12	1.50 ± 0.06	1.14 ± 0.03	1.39 ± 0.02	0.82 ± 0.01	0.69 ± 0.09	**0.58 ± 0.01**
	40	2.01 ± 0.03	1.53 ± 0.02	2.72 ± 0.09	1.53 ± 0.05	1.19 ± 0.03	1.37 ± 0.01	0.93 ± 0.02	–	**0.65 ± 0.01**
	50	1.95 ± 0.03	1.55 ± 0.02	2.91 ± 0.11	1.62 ± 0.04	1.17 ± 0.01	1.46 ± 0.01	0.99 ± 0.01	0.86 ± 0.02	**0.65 ± 0.01**
	100	1.75 ± 0.02	1.52 ± 0.01	3.02 ± 0.11	1.62 ± 0.03	1.15 ± 0.01	1.38 ± 0.01	1.19 ± 0.01	1.08 ± 0.03	**0.65 ± 0.00**
	200	1.60 ± 0.01	1.52 ± 0.01	2.94 ± 0.10	1.64 ± 0.02	1.15 ± 0.01	1.36 ± 0.01	1.29 ± 0.01	1.13 ± 0.04	**0.67 ± 0.00**

Table 8
Average best error before change ± standard error of CPSO, CPSOR, and CHPSO(ES-NDS) on DOPs with $d = 5$, $s = 1$ and varying number of peaks and change intervals.

Method	f	Number of peaks (m)								
		1	5	10	20	30	40	50	100	200
CPSO	500	67.88 ± 3.97	17.81 ± 0.49	15.65 ± 0.51	11.06 ± 0.42	8.92 ± 0.35	10.25 ± 0.36	9.26 ± 0.30	8.80 ± 0.29	7.09 ± 0.25
CPSOR		14.03 ± 1.46	12.00 ± 0.94	11.38 ± 0.93	10.55 ± 0.79	7.87 ± 0.59	10.04 ± 0.68	9.26 ± 0.56	9.43 ± 0.52	7.95 ± 0.41
CHPSO(ES-NDS)		**0.02 ± 0.003**	**1.83 ± 0.06**	**2.60 ± 0.05**	**2.81 ± 0.04**	**2.78 ± 0.03**	**2.74 ± 0.03**	**2.69 ± 0.02**	**2.60 ± 0.02**	**2.52 ± 0.02**
CPSO	1000	12.26 ± 0.46	6.97 ± 0.17	7.12 ± 0.19	5.33 ± 0.14	4.75 ± 0.13	5.20 ± 0.15	5.02 ± 0.13	4.69 ± 0.14	3.78 ± 0.12
CPSOR		1.41 ± 0.12	2.87 ± 0.14	2.98 ± 0.14	3.26 ± 0.12	2.88 ± 0.12	3.31 ± 0.14	3.72 ± 0.15	3.78 ± 0.14	3.49 ± 0.14
CHPSO(ES-NDS)		**5.90e−9 ± 3.02e−9**	**0.92 ± 0.04**	**1.60 ± 0.04**	**1.84 ± 0.03**	**1.83 ± 0.02**	**1.82 ± 0.02**	**1.81 ± 0.02**	**1.70 ± 0.01**	**1.63 ± 0.01**
CPSO	2500	4.40 ± 0.21	5.42 ± 0.11	3.39 ± 0.13	2.97 ± 0.09	2.67 ± 0.08	2.78 ± 0.09	2.82 ± 0.08	2.62 ± 0.08	2.04 ± 0.06
CPSOR		0.24 ± 0.03	1.11 ± 0.07	2.50 ± 0.07	1.59 ± 0.07	1.17 ± 0.07	1.43 ± 0.05	1.69 ± 0.07	1.78 ± 0.07	1.71 ± 0.07
CHPSO(ES-NDS)		**6.39e−13 ± 1.41e−13**	**0.39 ± 0.03**	**0.72 ± 0.03**	**0.94 ± 0.02**	**1.03 ± 0.02**	**1.03 ± 0.01**	**1.02 ± 0.01**	**0.99 ± 0.01**	**0.96 ± 0.00**
CPSO	5000	0.08 ± 0.02	0.80 ± 0.07	0.92 ± 0.09	1.43 ± 0.08	1.27 ± 0.07	1.31 ± 0.07	1.50 ± 0.07	1.48 ± 0.06	1.10 ± 0.05
CPSOR		0.04 ± 0.01	0.59 ± 0.05	**0.31 ± 0.03**	0.91 ± 0.06	**0.61 ± 0.05**	0.74 ± 0.04	0.97 ± 0.06	1.18 ± 0.06	1.05 ± 0.05
CHPSO(ES-NDS)		**2.58e−12 ± 1.77e−12**	**0.24 ± 0.02**	0.40 ± 0.02	**0.54 ± 0.01**	**0.61 ± 0.01**	**0.60 ± 0.01**	**0.60 ± 0.01**	**0.59 ± 0.00**	**0.60 ± 0.00**
CPSO	10,000	3.92e−5 ± 2.14e−4	0.69 ± 0.06	0.71 ± 0.09	1.11 ± 0.08	0.86 ± 0.06	0.87 ± 0.06	1.05 ± 0.06	1.01 ± 0.06	0.71 ± 0.04
CPSOR		2.44e−3 ± 7.09e−3	0.36 ± 0.04	**0.13 ± 0.02**	0.54 ± 0.04	**0.34 ± 0.03**	0.49 ± 0.03	0.65 ± 0.05	0.84 ± 0.05	0.75 ± 0.04
CHPSO(ES-NDS)		**6.03e−13 ± 8.80e−14**	**0.15 ± 0.01**	0.25 ± 0.01	**0.33 ± 0.01**	0.36 ± 0.01	**0.38 ± 0.01**	**0.38 ± 0.01**	**0.34 ± 0.00**	**0.33 ± 0.00**

Table 9
Average offline error ± standard error of algorithms on DOPs with $m = 10$, $d = 5$ and different shift lengths.

Method	Shift length (s)					
	0.0	1.0	2.0	3.0	4.0	5.0
mQSO	1.37 ± 0.06	1.77 ± 0.05	2.35 ± 0.04	2.90 ± 0.05	3.47 ± 0.06	3.95 ± 0.06
AmQSO	1.01 ± 0.05	1.64 ± 0.05	2.26 ± 0.05	2.86 ± 0.06	3.35 ± 0.06	3.80 ± 0.07
CellularPSO	0.93 ± 0.09	1.93 ± 0.08	2.72 ± 0.10	3.70 ± 0.11	4.71 ± 0.14	5.68 ± 0.18
mPSO	1.05 ± 0.09	1.66 ± 0.08	2.54 ± 0.11	3.46 ± 0.11	4.51 ± 0.15	5.32 ± 0.09
HmSO	1.03 ± 0.11	1.47 ± 0.04	2.54 ± 0.13	3.79 ± 0.16	5.03 ± 0.19	6.04 ± 0.22
TP-CPSO	0.48 ± 0.03	0.96 ± 0.05	1.25 ± 0.04	1.69 ± 0.04	2.12 ± 0.05	2.46 ± 0.05
FTMPSO	–	0.67 ± 0.04	1.20 ± 0.06	1.40 ± 0.09	–	**1.69** ± **0.07**
CHPSO(ES-NDS)	**0.46** ± **0.02**	**0.64** ± **0.02**	**0.92** ± **0.01**	**1.22** ± **0.02**	**1.50** ± **0.02**	1.82 ± 0.01

Table 10
Average best error before change ± standard error of CPSO, CPSOR, and CHPSO(ES-NDS) on DOPs with $m = 10$, $d = 5$ and different shift lengths.

Method	Shift length (s)					
	0.0	1.0	2.0	3.0	4.0	5.0
CPSO	0.69 ± 0.09	0.92 ± 0.09	1.10 ± 0.11	1.15 ± 0.11	1.19 ± 0.10	1.33 ± 0.10
CPSOR	1.10 ± 0.11	**0.31** ± **0.03**	0.53 ± 0.05	0.73 ± 0.06	0.99 ± 0.07	1.25 ± 0.09
CHPSO(ES-NDS)	**0.35** ± **0.01**	0.40 ± 0.02	**0.39** ± **0.01**	**0.42** ± **0.01**	**0.47** ± **0.01**	**0.50** ± **0.02**

Table 11
Average offline error ± standard error of algorithms on DOPs with $m = 10$, $s = 1$ and different dimensionalities.

Method	Number of dimensions (d)			
	5	10	20	50
mQSO	1.77 ± 0.05	4.38 ± 0.12	8.37 ± 0.16	25.57 ± 0.63
AmQSO	1.64 ± 0.05	3.80 ± 0.10	7.60 ± 0.14	26.79 ± 0.61
mPSO	1.66 ± 0.08	4.52 ± 0.15	12.52 ± 0.20	119.80 ± 3.21
HmSO	1.47 ± 0.04	4.63 ± 0.17	25.40 ± 0.32	66.25 ± 1.37
CHPSO(ES-NDS)	**0.64** ± **0.02**	**3.32** ± **0.06**	**5.04** ± **0.07**	**9.95** ± **0.11**

Considering Table 14, when the environment is unimodal, the best error before change for CHPSO(ES-NDS) is outstandingly better than the other contestants. For $m = 5$, 10, 20, and 30, the reported results by FTMPSO is slightly better than those of CHPSO(ES-NDS). However, as the number of the peaks in the landscape is increased, CHPSO(ES-NDS) can reach lower error than FTMPSO. From Table 14, in DOPs with many optima, i.e. $m = 100$ and 200, CHPSO(ES-NDS) is significantly superior to FTMPSO.

From the comparison results of CHPSO(ES-NDS) with the other contestant methods on the MPB problem with various

Table 12
Average best error before change ± standard error of CPSO, CPSOR, and CHPSO(ES-NDS) on DOPs with $m = 10$, $s = 1$ and different dimensionalities.

Method	Number of dimensions (d)			
	5	10	20	50
CPSO	0.92 ± 0.09	4.74 ± 0.14	9.84 ± 0.20	33.96 ± 0.70
CPSOR	**0.31** ± **0.03**	**2.70** ± **0.13**	5.31 ± 0.16	50.67 ± 6.02
CHPSO(ES-NDS)	0.40 ± 0.02	2.79 ± 0.06	**4.35** ± **0.08**	**7.94** ± **0.12**

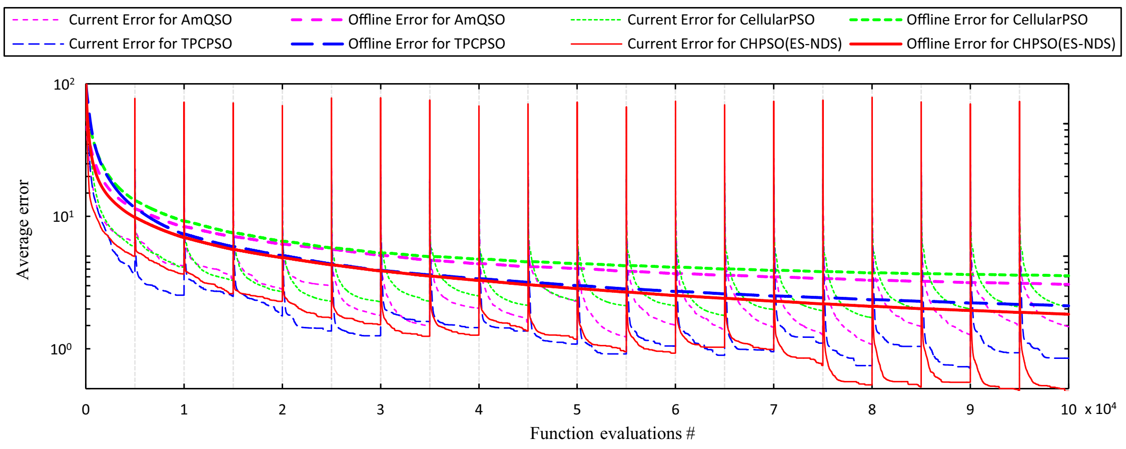

Fig. 2. Offline error and current error for different PSO variants in a dynamic environment with $m = 50$ and $f = 5000$. The vertical dotted lines in the figure indicate the start of a new period.

Table 13
Comparison of the average offline error ± standard error of the recent or well-known algorithms for dynamic environments modeled with MPB (Scenario 2).

Method	*m*								
	1	5	10	20	30	40	50	100	200
DynPopDE	–	1.03 ± 0.07	1.39 ± 0.03	–	–	–	2.10 ± 0.03	2.34 ± 0.02	2.44 ± 0.02
CDE	–	–	0.92 ± 0.03	–	–	–	–	–	–
DMAFSA	0.55 ± 0.06	0.78 ± 0.06	1.01 ± 0.05	1.42 ± 0.06	1.63 ± 0.06	–	1.84 ± 0.07	1.95 ± 0.05	1.99 ± 0.04
mNAFSA	0.38 ± 0.06	0.55 ± 0.04	0.90 ± 0.03	1.25 ± 0.06	1.47 ± 0.05	–	1.65 ± 0.05	1.83 ± 0.05	1.84 ± 0.05
CCPSO	**0.09** ± **0.00**	**0.25** ± **0.01**	0.75 ± 0.06	1.21 ± 0.08	1.40 ± 0.07	1.47 ± 0.08	1.50 ± 0.09	1.76 ± 0.09	–
CDEPSO	0.41 ± 0.00	0.97 ± 0.01	1.22 ± 0.01	1.54 ± 0.01	2.62 ± 0.01	–	2.20 ± 0.01	1.54 ± 0.01	2.11 ± 0.01
SFA	0.42 ± 0.07	0.89 ± 0.09	1.05 ± 0.04	1.48 ± 0.05	1.56 ± 0.06	–	1.87 ± 0.05	2.01 ± 0.04	1.99 ± 0.06
CHPSO(ES-NDS)	0.19 ± 0.00	0.44 ± 0.02	**0.64** ± **0.02**	**0.91** ± **0.01**	**0.99** ± **0.01**	**1.02** ± **0.01**	**1.03** ± **0.01**	**1.04** ± **0.01**	**1.01** ± **0.00**

Table 14
Comparison of the average best error before change ± standard error of the recent or well-known algorithms for dynamic environments modeled with MPB (Scenario 2).

Method	*m*								
	1	5	10	20	30	40	50	100	200
MPSO	–	0.56 ± 0.01	0.71 ± 0.03	1.11 ± 0.08	–	–	1.58 ± 0.10	–	–
CGAR	2.02 ± 0.01	2.56 ± 0.02	2.60 ± 0.02	3.66 ± 0.02	3.12 ± 0.02	–	3.26 ± 0.02	2.68 ± 0.01	2.39 ± 0.01
CDER	0.903 ± 0.04	8.02 ± 0.06	5.52 ± 0.03	7.49 ± 0.05	5.51 ± 0.02	–	5.79 ± 0.03	4.12 ± 0.02	3.71 ± 0.02
EM-POP	0.10 ± 8.50e−03	0.23 ± 0.02	0.65 ± 0.05	1.19 ± 0.08	1.42 ± 0.08	1.48 ± 0.09	1.49 ± 0.07	1.38 ± 0.10	1.14 ± 0.06
FTMPSO	3.48e−05 ± 9.01e−06	**0.20** ± **0.10**	**0.25** ± **0.05**	**0.42** ± **0.05**	**0.48** ± **0.04**	–	0.61 ± 0.03	0.83 ± 0.04	0.91 ± 0.06
CHPSO(ES-NDS)	**2.58e−12** ± **1.77e−12**	0.24 ± 0.02	0.40 ± 0.02	0.54 ± 0.01	0.61 ± 0.01	**0.60** ± **0.01**	**0.60** ± **0.01**	**0.59** ± **0.00**	**0.60** ± **0.00**

configurations, we can reach to the conclusion that the CHPSO(ES-NDS) algorithm is a highly effective method for optimization in dynamic environments.

There are several key features that contribute to the high performance of CPSOLS(ES-NDS) on the dynamic test environments which can be summarized as follows:

(1) *Small population size*. Many multi-population algorithms, e.g. mQSO [18], CellularPSO [24], CPSO [21], etc. are started from a big initial population. Therefore, these methods use too many function evaluations to detect the potential peaks. In contrast, CPSOLS(ES-NDS) starts with just three particles and dynamically adds and removes LSAs as required. This feature is beneficial specifically for fast changing dynamic optimization problems.
(2) *Fuzzy social-only PSO model*. The proposed fuzzy social-only PSO model improves the exploration ability of the PSO. More importantly, it enables PSO to operate with fewer number of particles, i.e. three particles in CPSOLS(ES-NDS).
(3) *Density control mechanism*. The density control mechanism provides the opportunity to avoid too many LSAs exploiting the same sub-area of the search space and hence save fitness evaluations.
(4) *Hibernation mechanism*. The hibernation mechanism used in LSAs prevents wasting precious fitness evaluations on unproductive LSAs. Therefore, more evaluations are saved and then allocated for searching other areas of the search space.
(5) *Strong local search ability*. The local search component of CPSOLS(ES-NDS) is a key factor in exploiting the promising sub-regions of the search and tracking the movement of peaks in the next environment.
(6) *Competition mechanism*. By choosing the best LSA to perform extra local search operation, the error value is reduced if the best LSA is on the highest peak. Using this mechanism, the best performing LSAs receive more fitness evaluations which is favorable to the performance of the CPSOLS(ES-NDS).

5. Conclusions and future works

A new hybrid model has been presented in this paper, called PSOLS, for dealing with DOPs. The main motivation behind the present study is to combine the desirable explorative features of PSO with exploitative features of local search methods. The basic algorithm has then further extended based on the following adaptations.

– The first adapted algorithm aims at allocating more function evaluations to the best performing local search agents in the search space. In this regard, the best performing LSA receives an extra round of local search at each iteration. This way, we expect that more function evaluations to be spent on most promising areas of the search space.
– Saving computational resources, i.e. function evaluations, and allocating them to the successful individuals of the algorithm can improve the performance of the search process. Therefore, the main goal of the second adaptation is to stop the search process in LSAs that consume resources without contributing much to the optimization.
– In the third adaptation, a combination of both mentioned adaptations are presented.

In order to validate the effectiveness of the proposed methods, a wide range of experiments were carried out to compare the proposed approach with various state-of-the-art algorithms on MPB. The experimental results confirm that the proposed hybrid approach is a very suitable tool for optimization in dynamic environments.

Several research directions can be pursued as future works. First, the values of the parameters of PSOLS were adjusted based on some preliminary experiments. Therefore, a comprehensive sensitivity analysis on the effect of parameters could be a direction for future research. Second, since change detection will not work in many real-world dynamic optimization problems, e.g. noisy dynamic environments, it is interesting to propose a new method without change detection. Third, it is also very valuable to apply the proposed approach to several real-world problems.

At the end, the authors would like to share the source code of PSOLS and will make it publicly available at http://ceit.aut.ac.ir/~meybodi/.

References

[1] C. Li, S. Yang, D.A. Pelta, Benchmark Generator for the IEEE WCCI-2012 Competition on Evolutionary Computation for Dynamic Optimization Problems, 2011.

[2] T. Blackwell, Particle swarm optimization in dynamic environments, in: S. Yang, Y.-S. Ong, Y. Jin (Eds.), Evolutionary Computation in Dynamic and Uncertain Environments, Springer, Berlin, Heidelberg, 2007, pp. 29–49.

[3] J. Kennedy, R. Eberhart, Particle swarm optimization, in: International Conference on Neural IEEE. Networks, 1995, pp. 1942–1948.

[4] S. Sun, J. Li, A two-swarm cooperative particle swarms optimization, Swarm Evol. Comput. 15 (2014) 1–18.

[5] B.-Y. Qu, P.N. Suganthan, S. Das, A distance-based locally informed particle swarm model for multimodal optimization, IEEE Trans. Evol. Comput. 17 (2013) 387–402.

[6] G. Fornarelli, A. Giaquinto, An unsupervised multi-swarm clustering technique for image segmentation, Swarm Evol. Comput. 11 (2013) 31–45.

[7] A.S. Rakitianskaia, A.P. Engelbrecht, Training feedforward neural networks with dynamic particle swarm optimisation, Swarm Intell. 6 (2012) 233–270.

[8] Y. Yan, L. Osadciw, Density estimation using a new dimension adaptive particle swarm optimization algorithm, Swarm Intell. 3 (2009) 275–301.

[9] N.C. Sahoo, S. Ganguly, D. Das, Multi-objective planning of electrical distribution systems incorporating sectionalizing switches and tie-lines using particle swarm optimization, Swarm Evol. Comput. 3 (2012) 15–32.

[10] A.A. Mousa, M.A. El-Shorbagy, W.F. Abd-El-Wahed, Local search based hybrid particle swarm optimization algorithm for multiobjective optimization, Swarm Evol. Comput. 3 (2012) 1–14.

[11] S. Alam, G. Dobbie, Y.S. Koh, P. Riddle, S. Ur Rehman, Research on particle swarm optimization based clustering: a systematic review of literature and techniques, Swarm Evol. Comput. 17 (2014) 1–13.

[12] R.C. Xiaohui Hu, Eberhart, Adaptive particle swarm optimization: detection and response to dynamic systems, in: Proceedings of the 2002 Congress on Evolutionary Computation, 2002, pp. 1666–1670.

[13] T.M. Blackwell, P. Bentley, Don't push me! Collision-avoiding swarms, in: Proceedings of the 2002 Congress on Evolutionary Computation, 2002, pp. 1691–1696.

[14] T.M. Blackwell, Dynamic search with charged, in: Genetic and Evolutionary Computation Conference, 2002, pp. 9–13.

[15] S. Janson, M. Middendorf, A hierarchical particle swarm optimizer for noisy and dynamic environments, Genet. Program Evol. Mach. 7 (2006) 329–354.

[16] C. Li, S. Yang, A general framework of multipopulation methods with clustering in undetectable dynamic environments, IEEE Trans. Evol. Comput. 16 (2012) 556–577.

[17] H. Wang, D. Wang, S. Yang, Triggered memory-based swarm optimization in dynamic environments, in: V Giacobini (Ed.), Applications of Evolutionary Computing, Springer, Berlin, Heidelberg, 2007, pp. 637–646.

[18] T. Blackwell, J. Branke, Multiswarms, exclusion, and anti-convergence in dynamic environments, IEEE Trans. Evol. Comput. 10 (2006) 459–472.

[19] U. Halder, S. Das, D. Maity, A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments, IEEE Trans. Cybern. 43 (2013) 881–897.

[20] D. Parrott, Li Xiaodong, Locating and tracking multiple dynamic optima by a particle swarm model using speciation, IEEE Trans. Evol. Comput. 10 (2006) 440–458.

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

[22] A. Nickabadi, M. Ebadzadeh, R. Safabakhsh, A competitive clustering particle swarm optimizer for dynamic optimization problems, Swarm Intell. 6 (2012) 177–206.

[23] A. Nickabadi, M.M. Ebadzadeh, R. Safabakhsh, A novel particle swarm optimization algorithm with adaptive inertia weight, Appl. Soft Comput. 11 (2011) 3658–3670.

[24] A. Hashemi, M.R. Meybodi, Cellular PSO: a PSO for dynamic environments, in: Z. Cai, Z. Li, Z. Kang, Y. Liu (Eds.), Advances in Computation and Intelligence, Springer, Berlin, Heidelberg, 2009, pp. 422–433.

[25] A.B. Hashemi, M.R. Meybodi, A multi-role cellular PSO for dynamic environments, in: 14th International CSI Computer Conference, 2009, pp. 412–417.

[26] A. Sharifi, V. Noroozi, M. Bashiri, A.B. Hashemi, M.R. Meybodi, Two phased cellular PSO: a new collaborative cellular algorithm for optimization in dynamic environments, in: IEEE Congress on Evolutionary Computation, 2012, pp. 1–8.

[27] M. Kamosi, A. Hashemi, M.R. Meybodi, New particle swarm optimization algorithm for dynamic environments, in: B. Panigrahi, S. Das, P. Suganthan, S. Dash (Eds.), Swarm, Evolutionary, and Memetic Computing, Springer, Berlin, Heidelberg, 2010, pp. 129–138.

[28] M. Kamosi, A.B. Hashemi, M.R. Meybodi, A hibernating multi-swarm optimization algorithm for dynamic environments, in: Nature and Biologically Inspired Computing (NaBIC), 2010 Second World Congress On, 2010, pp. 363–369.

[29] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, M.R. Meybodi, A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization, Appl. Soft Comput. 13 (2013) 2144–2158.

[30] L. Liu, S. Yang, D. Wang, Particle swarm optimization with composite particles in dynamic environments, IEEE Trans. Syst. Man Cybern. B: Cybern. 40 (2010) 1634–1648.

[31] J. Karimi, H. Nobahari, S.H. Pourtakdoust, A new hybrid approach for dynamic continuous optimization problems, Appl. Soft Comput. 12 (2012) 1158–1167.

[32] R.I. Lung, D. Dumitrescu, A collaborative model for tracking optima in dynamic environments, in: IEEE Congress on Evolutionary Computation, 2007, pp. 564–567.

[33] R. Lung, D. Dumitrescu, Evolutionary swarm cooperative optimization in dynamic environments, Nat Comput. 9 (2010) 83–94.

[34] J.K. Kordestani, A. Rezvanian, M. Meybodi, CDEPSO: a bi-population hybrid approach for dynamic optimization problems, Appl. Intell. 40 (4) (2014) 682–694.

[35] H. Wang, S. Yang, W.H. Ip, D. Wang, A particle swarm optimization based memetic algorithm for dynamic optimization problems, Nat Comput. 9 (2010) 703–725.

[36] H.-G. Beyer, H.-P. Schwefel, Evolution strategies – a comprehensive introduction, Nat. Comput. 1 (2002) 3–52.

[37] P. Novoa-Hernández, C. Corona, D. Pelta, Efficient multi-swarm PSO algorithms for dynamic environments, Memet. Comput. 3 (2011) 163–174.

[38] J. Branke, Memory enhanced evolutionary algorithms for changing optimization problems, in: Proceedings of the 1999 Congress on Evolutionary Computation, Vol. 3, Washington, DC, 1999, p. 1882.

[39] J. Branke, Evolutionary Optimization in Dynamic Environments, Kluwer Academic Publishers, 2002.

[40] J. Branke, H. Schmeck, Designing evolutionary algorithms for dynamic optimization problems, in: A. Ghosh, S. Tsutsui (Eds.), Advances in Evolutionary Computing, Springer, Berlin, Heidelberg, 2003, pp. 239–262.

[41] K. Trojanowski, Z. Michalewicz, Searching for optima in non-stationary environments, in: Proceedings of the 1999 Congress on Evolutionary Computation, 1999, pp. 1843–1850.

[42] T.T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: a survey of the state of the art, Swarm Evol. Comput. 6 (2012) 1–24.

[43] A.E. Ranginkaman, J. Kazemi Kordestani, A. Rezvanian, M.R. Meybodi, A note on the paper "A multi-population harmony search algorithm with external archive for dynamic optimization problems" by Turky and Abdullah, Inf. Sci. 288 (2014) 12–14.

[44] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. (1979) 65–70.

[45] T. Blackwell, J. Branke, X. Li, Particle swarms for dynamic optimization problems, in: C. Blum, D. Merkle (Eds.), Swarm Intelligence, Springer, Berlin, Heidelberg, 2008, pp. 193–217.

[46] I. Rezazadeh, M. Meybodi, A. Naebi, Adaptive particle swarm optimization algorithm for dynamic environments, in: Y. Tan, Y. Shi, Y. Chai, G. Wang (Eds.), Advances in Swarm Intelligence, Springer, Berlin, Heidelberg, 2011, pp. 120–129.

[47] M. Plessis, A. Engelbrecht, Differential evolution for dynamic environments with unknown numbers of optima, J. Glob. Optim. 55 (2013) 73–99.

[48] M.C. du Plessis, A.P. Engelbrecht, Using competitive population evaluation in a differential evolution algorithm for dynamic environments, Eur. J. Oper. Res. 218 (2012) 7–20.

[49] D. Yazdani, M.R. Akbarzadeh-Totonchi, B. Nasiri, M.R. Meybodi, A new artificial fish swarm algorithm for dynamic optimization problems, in: IEEE Congress on Evolutionary Computation, 2012, pp. 1–8.

[50] H. Wang, S. Yang, W.H. Ip, D. Wang, A memetic particle swarm optimisation algorithm for dynamic multi-modal optimisation problems, Int. J. Syst. Sci. 43 (2012) 1268–1283.

[51] B. Nasiri, M.R. Meybodi, Speciation based firefly algorithm for optimization in dynamic environments, Int. J. Artif. Intell. 8 (2012) 118–132.

[52] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, M. Meybodi, M. Akbarzadeh-Totonchi, mNAFSA: a novel approach for optimization in dynamic environments with global changes, Swarm Evol. Comput. 18 (2014) 38–53.

[53] A.M. Turky, S. Abdullah, A multi-population electromagnetic algorithm for dynamic optimisation problems, Appl. Soft Comput. 22 (2014) 474–482.