IUST
Iran University of Science and Technology

Iranian Journal of Electrical and Electronic Engineering

Journal Homepage: ijeee.iust.ac.ir

Research Paper

A Framework for Adapting Population-Based and Heuristic Algorithms for Dynamic Optimization Problems

S. M. Ejabati* and S. H. Zahiri*(C.A.)

Abstract: In this paper, a general framework was presented to boost heuristic optimization algorithms based on swarm intelligence from static to dynamic environments. Regarding the problems of dynamic optimization as opposed to static environments, evaluation function or constraints change in the time and hence place of optimization. The subject matter of the framework is based on the variability of the number of algorithm individuals and the creation of feasible subspaces appropriate to environmental conditions. Accordingly, to prevent early convergence along with the increasing speed of local search, the search space is divided with respect to the conditions of each moment into subspaces labeled as focused search area, and focused individuals are recruited to make search for it. Moreover, the structure of the design is in such a way that it often adapts itself to environmental condition, and there is no need to identify any change in the environment. The framework proposed for particle swarm optimization algorithm has been implemented as one of the most notable static optimization and a new optimization method referred to as ant lion optimizer. The results from moving peak benchmarks (MPB) indicated the good performance of the proposed framework for dynamic optimization. Furthermore, the positive performance of practices was assessed with respect to real-world issues, including clustering for dynamic data.

Keywords: Increase and Decrease Individuals, Dynamic Optimization Problems (DOPs), Local Search, Multi Swarm.

1 Introduction

A great number of optimization problems occur in the real world within dynamic environments, in that global optimization value change over time. For instance, we can refer to clustering problem when the values of sample change over time. The purpose of clustering is to assume the dataset postulated for spaces separated from one another, and is expressed with a clustering center. Once the values of data change, so does global optimization, which is the centers of clusters here, where the conventional algorithms that work in static environments are no more unable to find new optimization, and we need dynamic optimization algorithms. In a general case, engineering processes consists of uncertain and complex problem optimization, in that objective function, constraints, and some components of a problem change over time.

It is obvious that following the change in the foregoing problems, global optimization will undergo change. By considering this, the goal of dynamic optimization is to search for the path to the changes of optimization place over time, besides finding optimizations. This issue is addressed within static optimization.

Many of the dynamic algorithm have been developed by disclosing a change in the environment, where there are some techniques to disclose the change in the environment. Disclosing change in a workplace is difficult and eve it can be impossible at some time. Suppose that only a fraction of the entire search space receives change, in this case predicting this subspace or disclosing the change will be quite difficult. This is why methods that do not require change disclosure in an

Iranian Journal of Electrical and Electronic Engineering, 2020.
Paper first received 03 March 2019, revised 29 June 2019, and accepted 17 July 2019.
* The authors are with the Electrical and Computer Faculty, University of Birjand, Birjand, Iran.
E-mails: ejabati_masoud@birjand.ac.ir and hzahiri@birjand.ac.ir.
Corresponding Author: S. H. Zahiri.

environment can demonstrate better performance in a variety of environments.

Considering their dynamic nature, heuristic algorithms have a good potential to be used in a dynamic optimization. Since the optimization process of the algorithms is derived from a natural or biological evolution, and the nature is constantly changing, a variety of methods have been proposed recently to boost classic evolutionary algorithms that are used in static environments [1, 2]. The methods can be divided into five groups:

1) Increasing Diversity After a Change

Many strategies used to solve dynamic problems depend on the recognition of changes to the environment. Hence, this class of algorithms enhance the diversity through reinitiating the algorithm after discovering the change in the environment. Some part of the population is assigned a value at random when a change occurs in the environment [3, 4]. In these researches, a conventional particle swarm optimization (PSO) algorithm was applied on a time-varying parabolic function that reflects a simple unimodal fitness landscape.

1.2) Maintaining Diversity During Execution

In this class, the diversity is maintained during the time the program runs using different techniques. For example, Blackwell and Branke introduced two algorithms of multi-charged particle swarm optimization (mCPSO) and multi-quantum swarm optimization (mQSO) inspired by a nuclear field [5]. In mCPSO, some part of the swarm particle in each population is introduced as charged particles, which repel each other and revolve around the neutral particles in circles. The quantum particles move around the best particle in mQSO algorithm [6]. Both algorithms improved through re-initialization of the worst population when all swarms converge in order to maintain diversity. The same quantum principle was employed in the specification (SFA) algorithm [7].

A modified artificial bee colony (MABC) was proposed by Takano *et al*. with the restriction of communication among those bees that are closer only [8].

A PSO-based memetic algorithm was proposed by Wang *et al*. in which two local search operators were employed to maintain the diversity when implementing the algorithm [9]. Chen *et al*. employed a method similar to that of honeybee life cycle to adjust the colony size during optimization [10].

1.3) Memory Schemes

The memory schemes are used to transfer useful information from the search in the previous environment to the current one to speed up the search process. Usually, hopeful solutions are stored in the memory and reused when a change is made to the environment. An external archive is usually required to implement a memory scheme. The population archive is created to store the best solutions when a change is made to the environment, or when a peak is found [11]. Nasiri and Meybodi used two types of memories one of which used to store the information of current environment and the other one to store the information of the previous setting [12]. Such a mechanism has been also employed for PSO [13].

1.4) Multiple Population Methods

In the multi-population scheme, the search space is divided into some sub-spaces and each swarm is responsible for searching in one of the sub-spaces, which has multiple advantages [14], including maintenance of the global diversity since the start of the search at different areas, the possibility of tracking some optima at the same time, and facilitation of expanding the schemes based on single-swarm and multi-swarm approaches.

1.5) Hybridizations

Hybridizations are in fact a combination of methods with different knowledge domains such as genetic algorithms, differential evolution, or other meta-heuristic methods. For example, Multi-strategy ensemble PSO (MEPSO) uses Gaussian local search and differential mutation for exploration and exploitation [15]. A fuzzy cognition with multiple local searches was used by Sharifi *et al*. [16]. A hybrid model of cellular automata and PSO was also provided by Hashemi and Meybodi [17]. The search space was divided into discrete cells by automata and a particular number of particles was placed in each cell to maintain the diversity.

As mentioned earlier, the goal of dynamic optimization is to find and pursue the global optimization over time. Obviously, pursuing global optimization in a set of the best optimization has proved to be more functional when the environment changes [18, 19]. Accordingly, in many of the studies, the use of multi-population methods is suggested.

The principles of work are such that the entire search space is divided into subspaces. Each sub space covers one or a few local optimums and represents a sub-population. The algorithm then individually updates the particles of each sub-population and searches for optimum results. The challenging topic in multi-population methods is how to create the right number of sub-populations and that of people to cover different sub-domains in the search space. For example, the hierarchical clustering method [20, 21] was used to automatically divide the search space into sub-populations.

In this paper, a new framework with a simple yet robust mechanism has been introduced to boost heuristic algorithms based on swarm intelligence from optimization in static environment to dynamic environments. Similarly, new practices have been introduced for problems like how sub-populations can occur, preventing early convergence, and excessive swarm of individuals, and how individuals' behavior can adapt to the conditions of dynamic environments.

For performance evaluation, the algorithms PSO and AL have prepared for the proposed framework in order to solve the problems with dynamic environments.

The proposed algorithms are used to solve moving peak benchmark (MPB) problem with different settings [22]. Similarly, the method of optimization proposed to solve the problems in real world, e.g., clustering is used when samples are changing.

Organization of this paper is in a way that a review of multiple-population method developed for solving dynamic problems is presented in Section 2; the structure of the proposed framework and developed algorithms is introduced with a framework in Section 3. Section 4 includes the results obtained from algorithm performance when encountering MPB problem with different settings and dynamic clustering. In the end, section 5 ends with a conclusion and discussion.

2 Multi-Population Methods

In optimizing dynamic environments, preserving the diversity of population is one of the most fundamental issues. In the same vein, many research studies on multi modal problems in dynamic environments have shown that to preserve the diversity of population, the multi-population approach is highly effective.

A multi-colony ant colony optimization (ACO) algorithm was developed in which each colony uses a separate pheromone table to maximize exploration in the entire search space [24-26]. Although there is no explicit method for maintaining colonies in the entire search space, the results showed a better performance than single-clone algorithm.

Melo *et al*. used a multi-colony ACO algorithm with the difference that the pheromone table is similar for all clones called castes, and that the differences in the settings of parameters cause different behaviors of castes to cover more areas of search space [27], [28]. A similar idea was also proposed in the PSO [29]. Khouadjia *et al*. introduced a multi-population PSO algorithm based on the island model, so that particles migrate regularly between different populations [30]. According to Okulewicz and Mańdziuk, the algorithm functions differently, and communication between populations occurs only when changes occur in the environment [31].

Two groups of heterogeneous populations cooperate in collaborative evolutionary swarm optimization algorithm (CESO) [32]. The first group is responsible for maintaining diversity, and the second group undertakes tracking of global optimum. The groups follow crowding differential evolution (CDE) and PSO model respectively. Similarly, the binary population was employed by Zheng and Liu [33]. The number of populations exchanging information through evolutionary swarm cooperative algorithm reached three through adding a further population by Lung and Dumitrescu [34].

A self-organizing scout (SOS) algorithm was proposed by Branke *et al*. [35]. In the present research, a large main population (parents) is used to search for optima, and several small populations (children) have been employed to follow each optimum point. Whenever a new peak is found, a new population will be created for tracking. This approach was adopted in different types of EAs and meta-heuristics, e.g. GA [36], DE [32], and PSO [37]. An amendment GA algorithm was proposed for this purpose, i.e. making use of a great population to search and a small population to follow the changes [38]. The only difference that only two populations with the capability of overlapping have been employed in this algorithm.

The other approach is to combine both duties of searching and following in both populations such that each population can find and follow new solutions [39]. When a sub-population finds a new peak, it is divided into two sub-populations to ensure that each peak is followed by one sub-population.

In a number of studies, sub-populations are created through separation from the main population. For example, the fast multi-swarm optimization (FMSO) algorithm starts with a parent population [40]. When the best particle achieves a certain improvement criterion, the offspring population, which is a collection of the best particles, and also a number of its surrounding particles, are born. A similar idea viz. hibernation multi-swarm optimization algorithm (HmSO) was introduced by Kamosi *et al*. [41], in which the offspring population, if not effective, undergoes so-called hibernation until it feels a change in the environment.

The properties of atoms and the principle of repulsion of particles with the same name were used to preserve diversity in populations [5], [6]. In this way, charged populations are presented separately to cover optimizations. In an improved version, two innovative laws were added to increase diversity [42]. According to one of the rules, the number of quantum particles increases and that of particle paths decreases in case of an environmental change. Particles with inappropriate performances are initialized or stopped in the second law. Similar ideas of the exclusion principle were adopted in such other algorithms as multi-Bacterial foraging optimization (multi-BFO) [43], multi-swarm modified AFSO [44], and multi-swarm firefly algorithm (FA) [45].

The clustering PSO algorithm was provided by Yang and Li in which a hierarchical clustering method was used to divide the initial population into sub-populations to cover different local areas after recognizing a change to the environment [20, 21]. Then, a new version was introduced to avoid the recognition of an environmental change [46]. In this method, random particles are clustered in the new populations when the number of particles is less than a certain threshold value.

A decision-making model was proposed by Dun-wei and Xiao-yan in order to improve MPCEGAs on the

networks [47]. In this model, the objective of optimization is to minimize maximal computational time for average one-step iteration of a computational node. However, a limitation is considered to be the fact that each evolutionary population is allocated at most to one computational node. Accordingly, the computational resource can be allocated logically leading to a dramatic decrease in the implementation time on the computational nodes for the evolutionary populations.

A cooperative co-evolution strategy has been proposed based on the environmental sensitivity through analyzing the relationship between decision variables and environment [48]. According to this strategy, decision variables are divided into two subgroups namely the environmental subgroup and non-environmental subgroup, which are then optimized by two sub-populations, respectively. Therefore, the search space of each sub-population decreases and the algorithm capability of exploring improves. If environmental changes are examined during the evolution process, the sub-populations are adjusted based on the results of linear prediction and the Cauchy perturbation strategies.

3 Proposed Framework

In this section, a new design is introduced as increasing-decreasing optimization for optimization in dynamic environments. For the synchronization of heuristic algorithm based on swarm intelligence for dynamic environments, individuals are divided into two groups; free individuals and focused individuals. By individuals, we mean active components of optimization algorithm based on swarm intelligence in an optimization. For instance, in PSO algorithm noun as a particle in inclined planes system optimization (IPO) algorithm, ball or ant lion optimizer algorithm (ALO), ant is the active component of an optimization. The duty of free individuals is to always search throughout a searching space.

When these individuals advance toward convergence, some of them are referred to as focus individuals in a region called the focused search zone donated as FSZ. The area of FSZ is determined around the best focus individuals with a radius, called the focal search radius (FSR). By refreshing the focus individuals in case of improvement of the best individual, the FSZ center will also shifts to the location of the best individual, and other focus individuals in this FSZ will require to search in this subarea. At the same time, all free individuals are scattered to find other optimum points in the search space. In fact, the task of free individuals is to conduct an elementary search to find the probable subareas. Focus individuals function to find and improve the optimum and follow it during environmental changes. Thus, the number of individuals increases with respect to the creation of FSZs, and consequently the production of new focus individuals. The structure of the algorithm is such that at any time all focus individuals of that zone are eliminated depending on optimum environment conditions associated with each FSZ leading to constant changes in the number of individuals.

The main circle of the proposed plan begins with initialization of free individuals and then in the next phase the locations of these individuals are updated with a heuristic algorithm based on swarm intelligence. In what follows, free individuals' convergence is investigated. In the case of recognizing the possibility of convergence, individuals who are in a focused search zone (FSZ), (i.e., within the range of focused search radius that revolves around the best individual) are recognized as the focused individuals, creating a new search focused zone. Now the location of focused individuals in each search zone is separately updated with a heuristic algorithm. In the case of optimization improvement, any FSZ is transferred to the center of a better optimization. In the next phase, focused search zone is studied with respect to overlapping dimension and finally the FSZs that lost their optimization because of critical changes in the environmental conditions are deleted. It is because we assume that any change in an optimal location fall within FSZ area after being changed. Thus the total numbers of individuals are always a function of environmental conditions during operation, and this feature exempt the plan from constant updating constraints for the large number of individuals.

In the first step, free individuals are updated by a heuristic algorithm. In the next phase of the main circle of the proposed plan, the convergence process of free individuals will be investigated.

The initial convergence condition is that the rate of change in the position of the best free individual and its fitness is measured relative to the previous stage, followed by investigating the process of free individual convergence in Algorithm 1. If the best free individual changes are less than $r_{conv}/5$ and their fitness variations are less than r_{conv}, then the algorithm recognizes the possibility of free individual convergence. The position of the best point is examined in this stage. If it is located in each of the FSZs and its fitness value is more optimized than that of the center of the FSZ located therein, then the FSZ and, consequently, the focused member individuals are removed, and the new FSZ with the number of free individuals in the FSR range are formed with a centrality of the best free individual. Then, all free individuals are scattered throughout the search space to search for other possible areas. On the other hand, if the best free individual fitness value is not more optimal than the center of FSZ located therein, all of the free individuals would be scattered.

After satisfying the initial condition of convergence, if the position of the best free individual point is not located at any of the FSZs, then the total lengths of the

Algorithm 1 convergence Checking ().

```
1  if (f(x_gbest)^(t-1)-f(x_gbest)^t < r_conv) and (||(x_gbest)^(t-1)- (x_gbest)^t|| < r_conv/5) then
2      if x_gbest was within FSZ[i] then
3          if f(x_gbest) better than f(center_FSZ[i]) then
4              Replace selected free individual with focus individuals in FSZ[i]
5              re-initialize the free individuals
6          else
7              re-initialize the free individuals
8          end if
9      else
10         create a new FSZ with selected free individuals as the focus individuals
11         re-initialize the free individuals
12     end if
13 end if
```

two free individuals are discussed, which has the smallest distance from the best free individual. If this sum is also somewhat less than r_{conv}, it indicates that the free individuals are convergent, and best free individual is considered to be the center of FSZ. The number of free individuals up to a maximum of four individuals within the FSR range relative to the center are introduced as the focus individual. The title of its centralized individual is FSZ, and all free individuals are scattered throughout the search space to search for other possible areas. If there are less than two free individuals in the new FSZ range, two individuals closer to best free individual will be accepted as focus individuals for the mentioned FSZ.

In the next step, the position of focus individuals for each FSZ will be updated separately by the heuristic algorithm. If the best individual of each FSZ is improved, its center of FSZ will shift to the position of the best individual. In this way, the FSZs' location will also be updated in the search space. The framework also monitors the speed of focus individuals by scattering the mentioned focus individuals in their FSZ range when they are close to zero at any time, thereby, preventing their trapping in the local optimizations.

In the proposed framework, a mechanism has been used to prevent overlap of FSZs and the parallelization of focus individuals. In this method, the distances between FSZ centers are calculated and two neighboring FSZs are considered with a distance less than two times the FSR. In order to avoid the influence of local optimizations on the framework, average optimum performance is calculated by other FSZs. If the fitness of two FSZ centers were better than the average fitness (p_{mean}) of centers of other FSZs, the search radius would be reduced to half the distance between the centers of two FSZs in order for both FSZs to pursue their own optimums.

Now if the fitness of one or both FSZs was worse than the average fitness (p_{mean}) of the centers of other FSZs, the one whose optimum fitness was found to be worse than the other is considered as the local optimum. This

Algorithm 2 FSZ overlap Checking (*FSZ*,*FSR*).

```
1  if distance between FSZ[i] center and FSZ[j] center < 2*FSR then
2      f_mean(center_FSZ) = Calculation of mean fitness for other centers
3      if f(center_FSZ[i]) and f(center_FSZ[j]) better than P_mean×f_mean(center_FSZ) and FSR > 1 then
4          FSR = (distance between FSZ[i] center and FSZ[j] center) / 2
5      else
6          delete FSZ with worst fitness between FSZ[i] and FSZ[j]
7      end if
8  end if
```

Algorithm 3 Remove Useless FSZ.

```
1  f_mean(center_FSZ) = Calculation of mean all fitness centers
2  if worst f(center_FSZ[i]) > P_mean×f_mean(center_FSZ) then
3      delete FSZ with worst fitness FSZ[i]
4  end if
```

is followed by elimination of the mentioned FSZ and the associated focus individuals. This way prevents the search for several FSZs in a search area and also avoids trapping in the local optimizations resulting in an increase in the speed of the framework in finding the most optimum points in the entire search space within a more limited time. Algorithm 2 shows the FSZ overlap checking process.

Another idea used in the proposed framework presented in Algorithm 3 is the automatic removal of FSZs that do not have an optimum point in the region after severe changes in another environment, or the removal of focus individuals locally trapped in local optimizations. In this case, the coefficient of optimum optimization average (p_{mean}) is found as the measurement criterion. The worst-performing optimum in somewhat worse FSZs is compared with this criterion at each step, which eliminates the corresponding FSZ with its focus individuals.

3.1 Dynamic Environments With Undetectable Changes

Many studies carried out to solve dynamic optimization problems (DOPs) according to evolutionary algorithms are based on either detection of environmental changes [6, 20, 21, 34, 49], or prediction of a change assuming that such changes follow a pattern of properties [50]. When changes are detected or predicted, then different strategies are employed to increase diversity.

One of the common methods to detect a change is to re-evaluate the solutions. The algorithm re-evaluates regularly a number of solutions to recognize a change in the environment. Detectors can be a part of the population such as the best current optima [51], a memory-based sub-population [52], or a feasible sub-population [51]. In addition, the detectors can be selected separately from the searching population. For

example, they can be merely a fixed point (Carlisle and Dozier, 2000) [53] or a collection of random solutions [49].

Another detection idea is to supervise reductions in average values of the best points found during some iterations [54]. The changes that are based on statistical hypotheses tests are used to find the difference in the population distribution of two consecutive generations [49]. As stated by Morrison, studies on the detection of a change to the environment are based on the diversity, the relationship between diversity, value of fitness function, and the rate of success in detecting the change [55].

Given the importance of maintaining diversity in solving dynamic optimization problems and their dependence on environmental changes, it is important to note that performances of algorithms proposed for dynamic environments should not be dependent on the detection of changes in the environment, which are very difficult and impossible to discover in some cases. For example, imagine that changes occur in some areas of the search environment only making the algorithm very problematic to detect environmental changes. If it fails to detect, all common strategies will lose out deteriorating the algorithm. Also, detection of changes in noisy environments is very difficult for algorithms working based on detection of a change. The proposed algorithm does not need to detect a change in the environment and always adapts itself to environmental conditions.

3.2 Implementation of Framework on PSO Algorithm

In this section, a new algorithm called increasing-decreasing particle swarm optimization (idPSO) is introduced for optimization in dynamic environments. Due to its short and abstract mechanism, this algorithm has a good speed for dynamic optimization. The general framework is shown in Algorithm 4.

The main ring of the proposed algorithm starts with initialization for free particles, and in the next step, the free particle position is updated by particle swarm optimization algorithm, first proposed by Kennedy and Eberhart [56], [57]. Each particle i is represented by a velocity vector $\vec{v}_i$ and the position vector $\vec{x}_i$ updated by the version of Shi and Eberhart with the inertia weight as follows [58]:

$$v_i^{'d} = \omega v_i^d + c_1 r_1 \left(x_{pbest_i}^d - x_i^d \right) + c_2 r_2 \left(x_{gbest}^d - x_i^d \right) \quad (1)$$

$$x_i^{'d} = x_i^d + v_i^{'d} \quad (2)$$

In the above equation, $x_i^{'d}$ is the current position, x_i^d is the previous position, $v_i^{'d}$ is the current speed, and v_i^d is the previous speed of the i-th particle in the d dimension. $\vec{x}_{pbest_i}^d$ is the best position of the i-th particle so far, and $\vec{x}_{gbest_i}^d$ is the best position in the whole particles. $\omega \in (0, 1)$ is an inertia weight and determines the effect of the previous velocity on the current velocity of the particle; r_1 and r_2 are uniform random variables between zero and one; c_1 and c_2 are called acceleration coefficients and show the following of the particle of their best (cognitive) and the best collective (social component), respectively.

Given the various tasks for free particles and focus particles in idPSO, the acceleration coefficients and inertia weight for the particle update, which are responsible for finding probable areas in the entire search space and focus particle that are required to find and follow optimum on their own FSZ is considered different. The values of c_1, c_2, and ω of free particles for general search are considered to be 2, 2 and 0.4 and for focus particles with exact searches are considered as 3, 1 and 0.5, respectively.

In the next step, the convergence process of free particles will be studied. Then the location of focused particles of each FSZ is updated separately with a PSO algorithm. And in the case of an improvement of the best particle of each FSZ, the center of its FSZ will be updated in the search space.

In the following steps, FSZ's overlapping is studied, and then the mechanism for automatic deletion of additional FSZ is presented according to what has been laid down in the proposed framework.

Algorithm 4 idPSO.

```
1   Initialize free particles
2   while stop criteria is not satisfied do
3   for free particles do
4      PSO ();
5   end for
6   convergence checking (free particles)
7   if FSA exists then
8      for each FSZ[i] do
9         PSO(FSZ[i])
10        Update FSZ centers
11        Check velocity of focus particles
12     end for
13  end if
14  FSA overlap checking (FSZ, FSR)
15  Remove useless FSZ
16  end while
```

3.3 Implementation of Framework on ALO Algorithm

In this section, a new algorithm named as increasing-decreasing ant lion optimizer algorithm (idALO) is introduced for the optimization of dynamic environments (time-variable), as in what has been introduced in the proposed framework.

The Ant Lion optimizer is derived from the life cycle of these creatures in nature [58]. To trap ants, a cone-shaped cavity is created at the surface of the ground, and at the bottom of the cone waiting for the ant to fall into this cavity. When an ant falls into this cavity, it

tries to get out of the hole. At this time, the Ant Lion with the throw of sand to the top leads to the fall of the ant to the end of the cone and capture in the forks of the Ant Lion butter and finally the Ant Lion takes the prey into the ground and splits it. This algorithm is inspired by the interactions between Ant Lion and ant in the trap. Ants tend to move in the entire search space, and Ant Lions are allowed to hunt them and improve themselves.

Here ants are divided into two groups—free ants and focused ants—in order to make the ant lion algorithm prepared for solving optimization in dynamic environments. The duty of free ants is to constantly explore throughout the search space. When the ants are drifted toward a trap, a number of them fall within an area called focused search zone, and they are known focused ants. The focused search zone surrounding lion-ants is characterized by a radius called focused search radius. With each reiteration of algorithm, FSZ's center is changed into its location if the lion-ant improves. And other ants trapped in the zone are required to remain in this subspace. All free ants are scattered across the search space to find other optimal points. The pseudo-code idALO is shown in Algorithm 5. The process optimization proceeds according to the proposed framework.

4 Simulation Results

In this section, the dynamic algorithms derived from the proposed framework have undergone a test. This part can be divided into parts like investigation of algorithm mechanisms and analysis of key parameters in the process of algorithm optimization at the time of facing a MPB problem, analogy of algorithm performance with other methods set forth for solving MPB problem, and application of algorithms in the clustering when data are often changing.

4.1 Experimental Setup

1) MPB Problem is one of the most popular optimization problems in dynamic environments, which is widely used to evaluate dynamic optimization algorithms [22]. In an MPB issue, the optimum can be different with three features of position, height, and width of peaks. This problem is defined in D dimension as follows.

$$F\left(\vec{x},t\right)=\max_{i=1,\ldots,p}\frac{H_i\left(t\right)}{1+W_i\left(t\right)\sum_{j=1}^{D}\left(x_j\left(t\right)-X_{ij}\left(t\right)\right)^2} \tag{3}$$

In (3) $H_i(t)$ and $W_i(t)$ are the height and width of the peak i at time t, and $X_{ij}(t)$ is the j-th element of the peak location at time t. The p independently specifies the peaks blended together by the "max" function. The peak position in a random direction is transmitted by the vector $\vec{v}_i$ to a shift length of s, which represents the sensitivity of the problem's dynamics. The movement of a single peak is defined as the following relation:

$$\vec{v}_i\left(t\right)=\frac{s}{\left|\vec{r}+\vec{v}_i\left(t-1\right)\right|}\left(\left(1-\lambda\right)\vec{r}+\lambda\vec{v}_i\left(t-1\right)\right) \tag{4}$$

The transition vector $\vec{v}_i\left(t\right)$ is a linear combination of the random vector $\vec{r}$ and the previous transition vector $\vec{v}_i\left(t-1\right)$ normalized to the shift length of s. The value of the correlation parameter λ is zero, which indicates the non-alignment of the peak movements. Relationships of a peak change are expressed as:

$$H_i\left(t\right)=H_i\left(t-1\right)+height\_severity\times\sigma \tag{5}$$

$$W_i\left(t\right)=W_i\left(t-1\right)+width\_severity\times\sigma \tag{6}$$

$$\vec{X}_i\left(t\right)=\vec{X}_i\left(t\right)\left(t-1\right)+\vec{v}_i\left(t\right) \tag{7}$$

where σ is a random number with a normal distribution, a mean of zero and a variance of one.

2) Test Settings: Default setting of benchmark function in this study is shown in Table 1, similar to other algorithms presented for comparison with the proposed algorithm. The term "change of frequency U" means a change in the environment after U time evaluation of the fitness function. Peak location range is the range of peak changes in each dimension. The height of the peak varies randomly in the range of [30, 70] and the width changes in the interval [1, 12].

Algorithm 5 idALO.

```
Initialize the Free Ants
while stop criteria is not satisfied do
    for Free Ants do
        ALO ();
    end for
    Convergence Checking (Free Ants)
    if FSZ exist then
        for each FSZ[i] do
            ALO (FSZ [i])
            Update FSZ centers
        end for
    end if
    FSZ Overlap Checking (FSZ, FSR)
    Remove Useless FSZ
end while
```

Table 1 Default setting for MPB problem.

Parameter	Value
Number of peaks, p	[1,200]
Change frequency, U	5000
Height severity	7.0
Width severity	1.0
Peak shape	Cone
Basic function	No
Shift length, s	1.0
Number of dimensions, D	5
Correlation coefficient, λ	0
Peaks location range	[0,100]
Peak height, H	[30,70]
Peak width, W	[1,12]
Initial value of peaks	50.0

3) Performance Measurement: Several measurement methods have been introduced to measure the performance of algorithms in dynamic environments [59]. In order to create comparable results with other algorithms in this field, the offline error (OE) criterion is used, which is defined as the average difference in optimum value found by the algorithm with the global optimum value in each environment.

$$OE = \frac{1}{K}\sum_{k=1}^{K}\left(h_k - f_k\right) \tag{8}$$

In (8), f_k is the best answer found by the algorithm before the k-th change in the environment and h_k is the optimum value for the k-th environment. OE is mean difference of f_k and h_k in total K changes in the environment. All reported results represent more than 50 implementations of the program for 100 changes in the environment.

4.2 Sensitivity of the Algorithms to Structural Parameters

In order to investigate the sensitivity of the algorithm to the r_{conv} parameters (convergence radius), p_{mean} (average coefficient of fitness) and FSR (focus search radius) tests were performed on MPB with default values (p = 10, U = 5000, s = 1).

In fact, the offline error for the parameter changes at a change frequency of 5000 is approximately equal to and always reaches the smallest value. Thus, the effects of variations in each parameter have been shown based on average number of peaks found by the pattern.

As shown in Figs. 1 and 2, the effect of r_{conv} changes on the number of peaks, resulting from the random nature of the MPB problem in each replication, indicates that the algorithms do not show much sensitivity to r_{conv} changes. This is because an increase and/or decrease in r_{conv} actually reduce and/or raise, respectively, the time of transferring task of searching for possible areas and optimum point of free individuals to focus individuals. When r_{conv} rises, free individuals are detected earlier for convergence resulting in formation of a new FSZ, and begin to find optimum results, and free individuals return to their main task of finding probable areas. When it is possible to create local optimizations, this weakness is quickly modified by control algorithms through elimination of FSZ or enhancement of focus individuals. By contrast, by reducing r_{conv}, free individuals are in fact largely condemned to find the optimum amount in addition to searching for possible areas, which is among the tasks of focus individuals. This occurs when the speed of environmental changes increases (U variation frequency decreases). Here, due to the ability of the algorithms, r_{conv} changes have little effect on the amount of offline errors and the number of detected peaks.

The impact of p_{mean} changes of less than 0.7 on offline

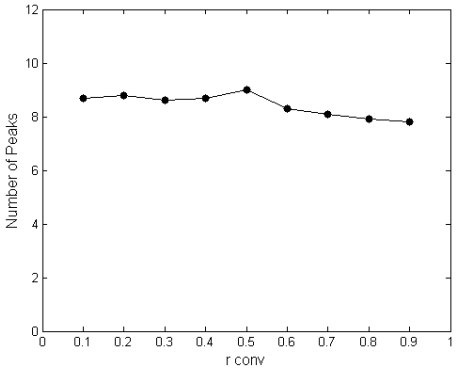

Fig. 1 The effect of the change in r_{conv} value on the mean of the number of peaks found in idPSO algorithm.

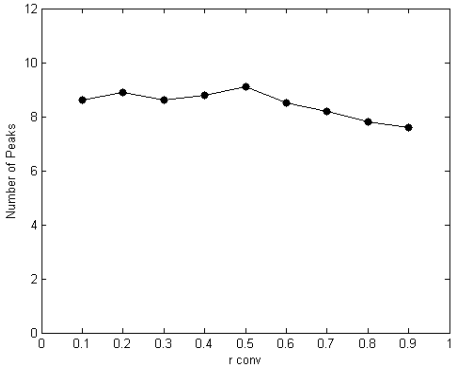

Fig. 2 The effect of the change in r_{conv} value on the mean of the number of peaks found in idALO algorithm.

errors is almost negligible. The process of the framework is in such a way that usually the optimum value is between the first four peaks. By increasing the amount of p_{mean} (Figs. 3 and 4), similar to what happens in r_{conv} changes, probable areas are generated by spending more time and delays until finding the peak. An increase of more than 0.7 would disrupt the trend of algorithms to find all possible areas and actually all the peaks. In case the first peaks found have high degrees of fitness, then they would have higher mean values and other control algorithms do not allow finding the peaks with minimum amounts of fitness. In such a case, it should be noted that the algorithm finds the best optimum point used in calculating offline errors. In the case of severe changes in the environment, however, it would not be possible to follow it by focus individuals, instead, it can be found by spending time again by free individuals in a new location. By contrast, with the reduction of p_{mean} to less than 0.3, the framework is confused with identifying the real probable areas of the local ones, and usually the number of FSZs exceeds the number of peaks (n = 10). This would be seen again in optimum global value and as a result, would not affect offline errors and just unnecessarily leads to increased number of individuals and consequently the amount of algorithm calculation. By definition, the FSR (focus search radius) determines the range of the focused search area around the best focus individuals of that

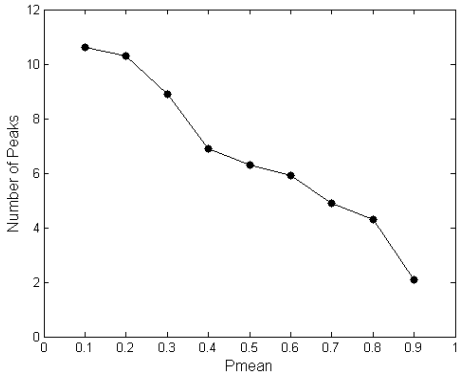

Fig. 3 The effect of the change in p_{mean} value on the mean of the number of peaks found in idPSO algorithm.

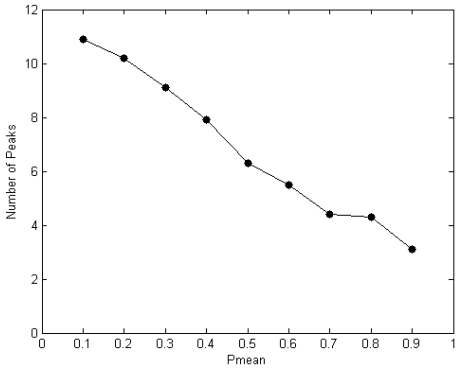

Fig. 4 The effect of the change in p_{mean} value on the mean of the number of peaks found in idALO algorithm.

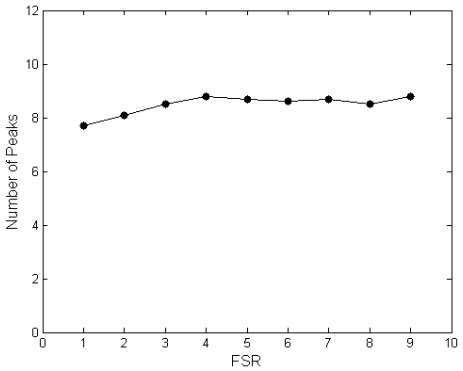

Fig. 5 The effect of the change in *FSR* value on the mean of the number of peaks found in idPSO algorithm.

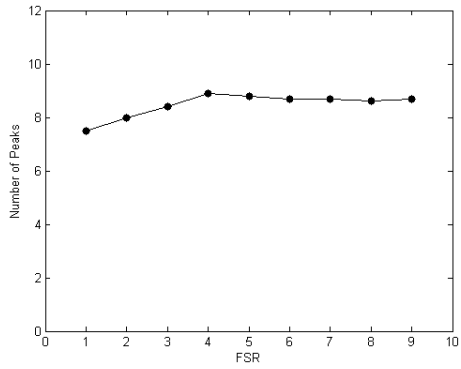

Fig. 6 The effect of the change in *FSR* value on the mean of the number of peaks found in idALO algorithm.

area in the entire optimization process. Two tasks are considered for this area. The first task is to find an optimum in the probable region and improve the optimum optimization until no change happens in the environment. The second task is when there are changes in the environment, which is followed by an optimum movement in such an environment. As a result, FSR value is important for the optimum to be in the same FSZ after environmental changes. Also, taking into account the large amount of FSR to ensure optimum change of location in the same area makes optimum improvement (first task) more difficult due to confronting with a larger area. Thus, there must be a compromise between optimum improvement and the presence of optimum after changes in the area. Figs. 5 and 6 show the effect of changing FSR values on the number of peaks found. Low FSR values (less than 2) improve the optimum as high as possible, but with a change in the environment, most FSZs lose their optimums and are eliminated by the *Remove Useless FSZ* mechanism. It is true that ultimately the proposed algorithm results in a great deal of time to optimize with frequency of environmental changes (U) of 5,000 and find the global optimum value with a high precision. Nonetheless, it is important that free individuals have to find new probable areas for creating FSZs and improving optimizations by new focused individuals meaning addition of extra calculation load and spending more time for optimization. As noted above, considering that high values (between 6 and 10) lead to an optimum improvement performed at a lower speed after a change in the environment, the probability of an optimum point would increase in this FSZ. In practice, high FSR values are mediated by the *FSZ overlap Checking* mechanism and reduce to mean values. The optimum values of parameters for solving the MPB problem are given in Table 2.

Table 2 Framework settings for solving the default MPB problem.

Parameter	Value
Number of Free Individuals	20
r_{conv}	0.5
p_{mean}	0.3
FSR	4

4.3 Comparison With Other Algorithms

In this part of the experiments, the proposed algorithms in terms of solving the MPB problem with different settings ware compared with other algorithms discussed in this topic, including CPSO [20], mCPSO [6], mQSO [6], CESO [32], SPSO [19], AmQSO [60], mPSO [41], APSO [61], FTMPSO [62], SFA [7], PSO-AQ [63], CDEPSO [64], rSPSO [65], CbDE-wCA [66], WD2O [67], BfCS-wVN [68], and cGA [69]. In the following, the effects of peak number

changes, shift severity, and frequency of environmental change U on offline errors of idPSO and idALO are provided in three sections.

1) Changes in the Number of Peaks: The series of experiments summarized in Table 3 indicate idPSO and idALO dealing with different number of peaks in the range 1 to 200 for the MPB problem. Comparisons were made based on offline errors and standard deviations with 17 algorithms derived from algorithms proposed in other studies with optimum settings. According to Table 3, the results of the proposed algorithm are better than other methods for the majority of peaks.

The algorithms settings for all peaks are the same as default settings. The accurate error values of proposed algorithms and other algorithms in Table 3 suggest that the amount of offline errors decrease with an increase in the number of peaks. It is more difficult for the algorithm to work on finding and tracking a high number of peaks. Though, it is logical that the closeness of the fitness values of these optimums to the fitness value of global optimum possibly increases with an increase in the found local optimums. Also, more regions are monitored and searched by focus individuals in the search space, which as a result, the possibility of optimum global presence in these areas would increase more rapidly after changes in the environment.

Given Fig. 7, a bar chart shows a difference in the values of offline error for the two proposed algorithms and other three algorithms within the changing frequency 5000 and the number of different peaks.

2) Changes in Shift Severity: Table 4 presents the offline error values of six similar algorithms and proposed algorithms for four different shift severity. The environment settings are the same change frequencies of 5000 for 10 peaks and shift severity values are 1, 2, 3, and 5, respectively. Obviously, increasing amount of shift severity results in elevated intensity of changes in the environment. In other words, a higher increase in the shift severity moves the optimum location to a further distance after a change in the environment rendering the algorithm with much more difficulty in tracking the optimum. Hence, the values of offline errors for the algorithms (Table 4) increase with rising shift severity values. Increased amount of offline errors for the proposed algorithms with the above-mentioned optimum settings can be ignored, and this point is an indicator of a high robustness of the proposed framework for finding and tracking an optimum for each shift severity.

Table 3 Comparison of offline error algorithms for different peak numbers in the MPB problem at the change frequency of U=5000.

Algorithm	Number of peaks, p							
	1	5	10	20	30	50	100	200
CPSO	0.14(0.11)	0.72(0.30)	1.06(0.24)	1.59(0.22)	1.58(0.17)	1.54(0.12)	1.41(0.08)	1.24(0.06)
mCPSO	4.93(0.17)	2.07(0.08)	2.08(0.07)	2.64(0.07)	2.63(0.08)	2.65(0.06)	2.49(0.04)	2.44(0.04)
mQSO(5,5q)	2.24(0.05)	1.82(0.08)	1.85(0.08)	2.48(0.09)	2.51(0.10)	2.53(0.08)	2.35(0.06)	2.24(0.05)
CESO	1.04(0.00)	-	1.38(0.02)	1.72(0.02)	1.24(0.01)	1.45(0.01)	1.28(0.02)	-
rSPSO	1.42(0.06)	1.04(0.03)	1.50(0.08)	2.20(0.07)	2.62(0.07)	2.72(0.08)	2.93(0.06)	2.79(0.05)
SPSO	2.64(0.10)	2.15(0.07)	2.51(0.09)	3.21(0.07)	3.64(0.07)	3.86(0.08)	4.01(0.07)	3.82(0.05)
AmQSO	2.62(0.10)	1.01(0.09)	1.51(0.10)	2.00(0.15)	2.19(0.17)	2.43(0.13)	2.68(0.12)	2.62(0.10)
mPSO	2.42(0.05)	1.82(0.08)	1.85(0.08)	2.48(0.09)	2.51(0.10)	2.53(0.08)	2.35(0.06)	2.24(0.05)
APSO	0.53(0.01)	1.05(0.06)	1.31(0.03)	1.69(0.05)	1.78(0.02)	1.95(0.02)	1.95(0.01)	1.90(0.01)
FTMPSO	0.18(0.01)	0.47(0.05)	0.67(0.04)	0.93(0.04)	1.14(0.04)	1.32(0.04)	1.61(0.03)	1.67(0.03)
SFA	0.42(0.03)	0.89(0.07)	1.05(0.04)	1.48(0.05)	1.56(0.06)	1.87(0.05)	2.01(0.04)	1.99(0.06)
PSO-AQ	0.34(0.02)	0.80(0.12)	0.89(0.03)	1.45(0.06)	1.52(0.04)	1.77(0.05)	1.95(0.05)	1.96(0.04)
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)
CbDE-wCA	0.14(0.03)	**0.30(0.02)**	0.86(0.08)	0.98(0.05)	1.34(0.04)	1.31(0.04)	1.35(0.03)	1.29(0.02)
WD2O	1.21(0.03)	0.76(0.003)	1.25(0.02)	1.22(0.01)	1.75(0.01)	1.87(0.01)	2.10(0.01)	1.95(0.01)
cGA	0.92(0.09)	1.06(0.07)	1.15(0.10)	1.18(0.06)	1.35(0.51)	1.65(0.07)	1.80(0.06)	1.71(0.05)
BfCS-wVN	0.30(0.06)	0.38(0.21)	**0.51(0.11)**	**0.74(0.12)**	1.00(0.11)	0.84(0.06)	1.11(0.06)	1.18(0.08)
idPSO	0.12(0.02)e-7	0.58(0.07)	0.62(0.05)	0.75(0.03)	0.67(0.04)	0.64(0.01)	0.25(0.02)	0.26(0.04)
idALO	0.09(0.02)	0.49(0.06)	0.71(0.06)	1.41(0.05)	1.30(0.07)	1.21(0.08)	1.20(0.09)	1.22(0.05)

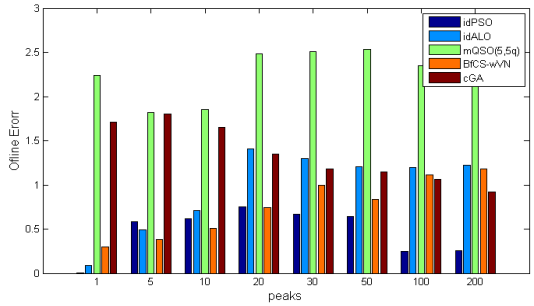

Fig. 7 Comparison of the algorithms of the proposed framework and other algorithms in a frequency of 5000 and with respect to the number of different peaks.

Table 4 Comparison of the offline error of algorithms for various shift Severity in MPB problem.

Algorithm	Shift severity, s			
	1	2	3	5
CPSO	1.06(0.24)	1.17(0.22)	1.36(0.28)	1.58(0.32)
mCPSO	2.05(0.07)	2.80(0.07)	3.57(0.08)	4.89(0.11)
mQSO(5,5q)	1.85(0.08)	2.40(0.06)	3.00(0.06)	4.24(0.10)
CESO	1.38(0.02)	1.78(0.02)	2.03(0.03)	2.52(0.06)
cGA	0.92(0.09)	2.19(0.15)	3.31(0.25)	6.45(0.45)
BfCS-wVN	**0.51(0.11)**	0.89(0.16)	1.26(0.13)	2.39(0.20)
idPSO	0.62(0.05)	0.67(0.07)	0.76(0.06)	0.91(0.09)
idALO	0.71(0.06)	0.81(0.07)	0.93(0.07)	1.09(0.08)

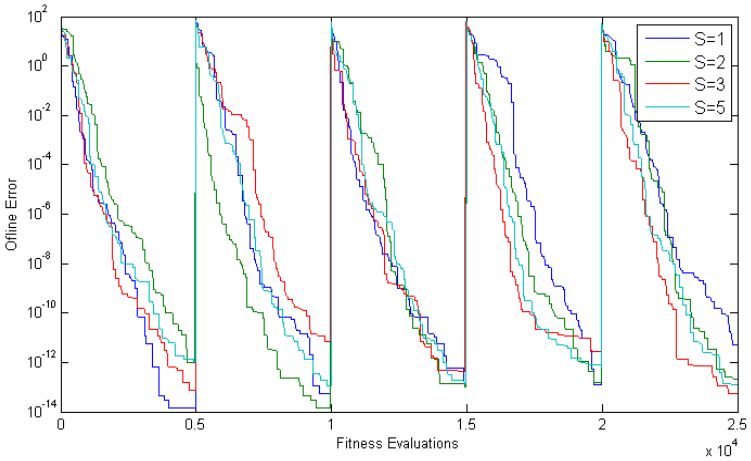

Fig. 8 The value of the offline error of the algorithm idPSO for the degree of a different change in frequency of 5000.

Table 5 Comparison of offline error algorithms for different peak numbers in the MPB problem at the change frequency of *U*=1000.

Algorithm	Number of peaks, *p*							
	1	5	10	20	30	50	100	200
mQSO(5,5q)	18.60(1.63)	6.56(0.38)	5.71(0.22)	5.85(0.15)	5.81(0.15)	5.87(0.13)	5.83(0.13)	5.54(0.11)
AmQSO	2.33(0.31)	2.90(0.32)	4.56(0.40)	5.36(0.47)	5.20(0.38)	6.06(0.14)	4.77(0.45)	5.75(0.26)
mPSO	4.44(0.02)	3.93(0.16)	4.57(0.18)	4.97(0.13)	5.15(0.12)	5.33(0.10)	5.60(0.09)	5.78(0.09)
APSO	2.72(0.04)	2.99(0.09)	3.87(0.08)	4.13(0.06)	4.12(0.04)	4.11(0.03)	4.26(0.04)	4.21(0.02)
FTMPSO	0.89(0.05)	1.70(0.10)	2.36(0.09)	3.01(0.12)	3.06(0.10)	3.29(0.10)	3.63(0.09)	3.74(0.09)
SFA	2.45(0.12)	2.71(0.06)	3.64(0.04)	4.01(0.07)	4.02(0.08)	4.12(0.07)	4.40(0.07)	4.43(0.07)
cGA	1.10(0.10)	**1.12(0.11)**	**1.28(0.13)**	1.76(0.09)	2.01(0.14)	2.56(0.10)	2.42(0.14)	2.20(0.11)
idPSO	0.21(0.02)	1.33(0.05)	1.61(0.03)	1.73(0.06)	1.70(0.04)	1.41(0.01)	0.92(0.03)	0.96(0.03)
idALO	1.05(0.06)	1.77(0.08)	2.12(0.08)	2.71(0.05)	2.96(0.06)	3.11(0.11)	3.21(0.08)	3.19(0.05)

Table 6 Comparison of offline error algorithms for different peak numbers in the MPB problem at the change frequency of *U*=10000.

Algorithm	Number of peaks, *p*							
	1	5	10	20	30	50	100	200
mQSO(5,5q)	1.90(0.18)	1.03(0.06)	1.10(0.07)	1.84(0.09)	2.00(0.09)	1.99(0.07)	1.85(0.05)	1.71(0.04)
AmQSO	0.19(0.02)	0.45(0.04)	0.76(0.06)	1.28(0.12)	1.78(0.09)	1.55(0.08)	1.89(0.14)	2.52(0.10)
mPSO	0.27(0.02)	0.70(0.10)	0.97(0.04)	1.34(0.08)	1.43(0.05)	1.47(0.04)	1.50(0.03)	1.48(0.02)
APSO	0.25(0.01)	0.57(0.03)	0.82(0.02)	1.23(0.02)	1.39(0.02)	1.46(0.01)	1.38(0.01)	1.36(0.01)
FTMPSO	0.09(0.00)	0.31(0.04)	0.43(0.03)	0.56(0.01)	0.69(0.09)	0.86(0.02)	1.08(0.03)	1.13(0.04)
SFA	0.26(0.03)	0.53(0.04)	0.72(0.02)	0.91(0.03)	0.99(0.04)	1.19(0.04)	1.44(0.04)	1.52(0.03)
BfCS-wVN	0.18(0.04)	0.20(0.02)	**0.30(0.06)**	0.84(0.05)	0.66(0.08)	0.54(0.70)	0.85(0.05)	0.85(0.05)
idPSO	0.3(0.04)e-12	0.12(0.03)	0.33(0.04)	0.44(0.07)	0.53(0.10)	0.43(0.05)	0.19(0.00)	0.19(0.00)
idALO	0.01(0.00)	0.31(0.03)	0.43(0.05)	0.58(0.03)	0.75(0.06)	0.77(0.04)	0.79(0.04)	0.77(0.03)

To confirm this, in Fig. 8 the value of offline error for the algorithm idPSO regarding the degree of various changes is based on the number of the request made. As can be seen, the effects of change on the degree of change in the behavior of algorithm during optimization are infinitesimal.

3) Changes in the Frequency of Environment Changes: Changes in the frequency of environmental changes U actually determines the time spent by the algorithm to find an optimum in each environment before occurrence of a change. Obviously, the lower the amount of frequency, the lesser the time dedicated to finding a new environment. On the other hand, the suitable opportunity to find and optimize the best optimum corresponds to the high frequency value. The results of MPB problem with various numbers of peaks and environmental change frequencies of 1000 and 10,000 are shown in Tables 5 and 6. For example, for idPSO algorithm, for 10 peaks at a change frequency of 1000, the error rate was 1.61, while at a frequency of 10,000, the algorithm needs more time, hence, the offline error reduced to 0.33. Other values for idPSO algorithm are better than the other algorithms at both frequencies.

In Figs. 9 and 10, the offline error of both algorithms derived from the proposed framework and other algorithms in the number of a different peak for a frequency of environment change is 1000 and 10000 respectively.

4.4 Application of the Proposed Framework in Clustering

In this section, the proposed algorithms for the

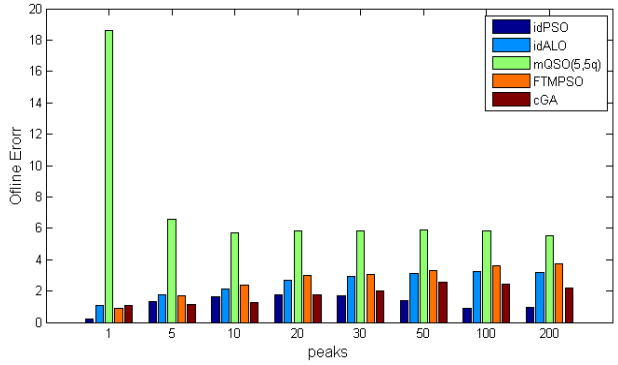

Fig. 9 Comparison of the algorithms of the proposed framework and other algorithms in a frequency of 1000 and with respect to the number of different peaks.

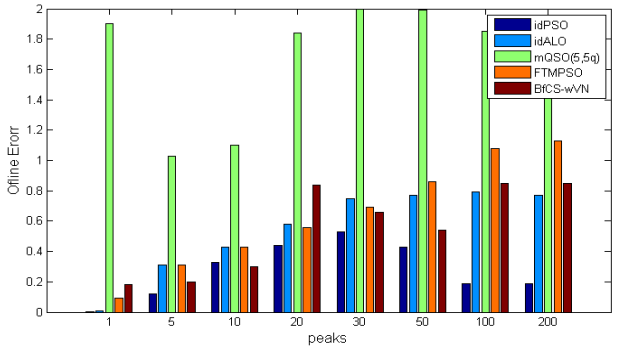

Fig. 10 Comparison of the algorithms of the proposed framework and other algorithms in a frequency of 10000 and with respect to the number of different peaks.

clustering of data whose values change over time are addressed. The difference between this part and previous part revolving around solving MPB problem is the definition of fitness function for algorithms. The fitness function is based on quantized error and the equation 9. Similarly, the structure of individuals is defined with the equation 10, which includes the centers of clusters. In what follows, since there is no database with samples varying at a valid time, data of each class are generated with different mean (μ) and standard deviation (σ). The range of search area [0-50], and change frequency in the data location is estimated to be 100. A change frequency of 100 suggests that the algorithm of data location changes after every 100 times fitness assessment.

$$\text{fitness}(p)=\frac{1}{N}\sum_{j=1}^{k}\sum_{\forall X\in C_j} d\left(X,Z_j\right) \tag{9}$$

$$\underbrace{z_1^1\ z_1^2\ \cdots\ z_1^D}_{Z_1}\ldots\underbrace{z_k^1\ z_k^2\ \cdots\ z_k^D}_{Z_k} \tag{10}$$

In what follows, we deal with the algorithms introduced in the different conditions of data and environment. Since the samples generated with the information provided, and we know the values of the mean of samples of each class, these values are taken as the real centers of each cluster. The assessment criterion of the algorithm performance is similar to the measurement criterion for MPB problem, i.e. the mean of the distances of real centers of clusters with the centers found with the algorithm for 20 times change in the environment, and calculated according to (11).

$$OE=\frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K} d\left(Z_{m,k},Center_{m,k}\right) \tag{11}$$

In the above equation $Z_{m,k}$, the best centers found by the algorithm is somewhere before reaching m-th change in the environment and $Center_{m,k}$ is the real centers of clusters in m-th environment. OE is the mean of Euclidian distance $Z_{n,k}$ and $Center_{n,k}$ for the whole M change in the environment.

In Table 7, the proposed algorithms were compared for the change frequencies of 100 and 200, and the number of different clusters. It should be noted that the dimensions of the problem are summed up in four dimensions, and the number of samples in each class is 500. It is evident that as the number of clusters increases, so does the value of the error obtained. Moreover, as the frequency goes up to 200, the time of algorithm for the improvement of the centers found increases, and, in consequence, the values of the error decrease.

In Table 8, the analogy of the proposed algorithm error is presented in different dimensions and four clusters. As can be expected, as the number of data dimension increases, so does the error of algorithms. This increase in error can be somewhat modified as the change frequency increases.

Table 7 Comparison of the proposed algorithms for the number of different clusters in a 4-dimensional environment.

Change frequency	clusters	idPSO	idALO
100	2	0.012	0.019
	4	0.022	0.031
	6	0.142	0.202
	8	0.467	0.759
200	2	0.0056	0.0082
	4	0.0073	0.0105
	6	0.0693	0.0817
	8	0.348	0.491

Table 8 Comparison of the proposed algorithms for the different dimensions and the number of 4 clusters.

change frequency	dimensions	idPSO	idALO
100	2	0.0013	0.0018
	4	0.022	0.031
	6	0.252	0.381
	8	0.782	0.853
200	2	0.0008	0.0009
	4	0.0073	0.0105
	6	0.331	0.397
	8	0.446	0.572

5 Conclusion and Discussion

In this article, a general framework is presented to boost the heuristic optimization algorithms based on

swarm intelligence from static to dynamic environments. The main idea of the framework is based on the variability of the number of algorithm individuals and formation of possible subspaces suitable for environmental conditions.

In dynamic optimization, it is important to reduce the optimization time, in other words, a faster convergence to optimize while maintaining diversity throughout the search space. Most dynamic algorithms start with a large number of individuals, and the number of individuals is reduced with the advent of optimization processes. It is a waste of time to calculate the fitness of a high number of individuals in the early stages of optimization. In order to avoid this, proposed framework starts with low individuals, and always an increase in the number of individuals is a function of environmental condition, including increasing number of optimum points. In contrast, the algorithm attempts to reduce the number of individuals according to the environmental conditions, which in turn reduces the calculation load and increases the speed of optimization. Environmental changes are not always detectable, as an example, only a fraction of the total search area may change, or changes in noisy environments cannot be easily detected. In this case, the performances of algorithms based on the detection of environmental changes become a major problem. To overcome this shortcoming, proposed framework is designed with no need to detect changes in the environment and always adapts itself to environmental conditions. The proposed algorithms ware investigated regarding its efficiency for solving MPB problem as one of the most popular benchmark functions in dynamic environments. The range of experiments for different settings of the MPB problem, including number of various peaks, changes in shift severity, and frequency of different environmental changes, show the proper performance of the proposed algorithm in comparison with other dynamic optimization algorithms. In what follows, the introduced algorithms for the clustering of dynamic data whose values change over time were used, and we obtained positive results.

Future research suggestions include the use of an algorithm to optimize real world problems. It is also recommended to use the algorithm for dynamic clustering such as web data. The use of adaptive and self-adaptive mechanisms for structural parameters of the algorithm can result in more rapid adaptability of the method to environmental conditions.

References

[1] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: algorithms and applications," *Swarm and Evolutionary Computation*, Vol. 33, pp. 1–17, 2017.

[2] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, Vol. 6, pp. 1–24, 2012.

[3] R. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. 1, pp. 94–100, 2001.

[4] X. Hu and R. Eberhart, "Adaptive particle swarm optimization: detection and response to dynamic systems," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'02)*, pp. 1666–1670, 2002.

[5] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments. Applications of evolutionary computing," in *Workshops on Applications of Evolutionary Computation*, Springer, Berlin, Heidelberg, pp. 489–500, 2004.

[6] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Transactions on Evolutionary Computation*, Vol. 10, pp.459–472, 2006.

[7] B. Nasiri and M. Meybodi, "Speciation based firefly algorithm for optimization in dynamic environments," *International Journal of Artificial Intelligence*, Vol. 8, pp. 118–132, 2012.

[8] R. Takano, T. Harada, H. Sato, and K. Takadama, "Artificial bee colony algorithm based on local information sharing in dynamic environment," in *Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems*, Vol. 1, Springer, Cham, pp. 627–641, 2015.

[9] H. Wang, S. Yang, W. Ip, and D. Wang, "A memetic particle swarm optimization algorithm for dynamic multi-modal optimisation problems," *International Journal of Systems Science*, Vol. 43, No. 7, pp. 1268–1283, 2012.

[10] H. Chen, L. Ma, M. He, X. Wang, X. Liang, L. Sun, and M. Huang, "Artificial bee colony optimizer based on bee life-cycle for stationary and dynamic optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 47, No. 2, pp. 327–346, 2016.

[11] H. Wang, D. Wang, and S. Yang, "Triggered memory-based swarm optimization in dynamic environments," in *Workshops on Applications of Evolutionary Computation*, Springer, Berlin Heidelberg, pp 637–646, 2007.

[12] B. Nasiri and M. Meybodi, "History-driven firefly algorithm for optimization in dynamic and uncertain environments," *International Journal of Bio-Inspired Computation*, Vol. 8, pp. 326–339, 2016.

[13] B. Nasiri, M. Meybodi, and M. Ebadzadeh, "History-driven particle swarm optimization in dynamic and uncertain environments," *Neurocomputing*, Vol. 172, pp. 356–370, 2016.

[14] C. Li, T. T. Nguyen, M. Yang, S. Yang, and S. Zeng, "Multi-population methods in unconstrained continuous dynamic environments: The challenges," *Information Sciences*, Vol. 296, pp. 95–118, 2015.

[15] W. Du and B. Li, "Multi-strategy ensemble particle swarm optimization for dynamic optimization," *Information Sciences*, Vol. 178, pp. 3096–3109, 2008.

[16] A. Sharifi, J. K. Kordestani, M. Mahdaviani, and M. R. Meybodi, "A novel hybrid adaptive collaborative approach based on particle swarm optimization and local search for dynamic optimization problems," *Applied Soft Computing*, Vol. 32, pp. 432–448, 2015.

[17] A. Hashemi and M. Meybodi, "Cellular PSO: A PSO for dynamic environments," in *International Symposium on Intelligence Computation and Applications*, Springer, Berlin Heidelberg, Vol. 5821, pp. 422–433, 2009.

[18] L. Liu, S. R. Ranjithan, and G. Mahinthakumar, "Contamination source identification in water distribution systems using an adaptive dynamic optimization procedure," *Journal of Water Resources Planning and Management*, Vol. 137, pp.183–192, 2010.

[19] D. Parrott and X. Li, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," in *IEEE Transactions on Evolutionary Computation*, Vol. 10, pp. 440–458, 2006.

[20] C. Li and S. Yang, "A clustering particle swarm optimizer for dynamic optimization," in *IEEE Congress on Evolutionary Computation (CEC)*, pp. 439–446, 2009.

[21] S. Yang and C. Li, "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments," *IEEE Transactions on Evolutionary Computation*, Vol. 14, pp. 959–974, 2010.

[22] J. Branke, "The moving peaks benchmark," 1999. [Online]. Available: http://people.aifb.kit.edu/jbr/MovPeaks/movpeaks.

[23] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *IEEE Transactions on Evolutionary Computation*, Vol. 9, pp. 303–317, 2005.

[24] M. Mavrovouniotis and S. Yang, "Ant colony optimization with self-adaptive evaporation rate in dynamic environments," in *IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)*, pp. 47–54, 2014.

[25] B. Van Veen, M. Emmerich, Z. Yang, T. Bäck, and J. Kok, "Ant colony algorithms for the dynamic vehicle routing problem with time windows, in *International Work-Conference on the Interplay Between Natural and Artificial Computation*, Springer, pp. 1–10, 2013.

[26] Z. Yang, M. Emmerich, and T. Bäck, "Ant based solver for dynamic vehicle routing problem with time windows and multiple priorities," in *IEEE Congress on Evolutionary Computation (CEC)*, pp. 2813–2819, 2015.

[27] L. Melo, F. Pereira, and E. Costa, "Multi-caste ant colony algorithm for the dynamic traveling salesperson problem," in *International Conference on Adaptive and Natural Computing Algorithms*, Springer, pp. 179–188, 2013.

[28] L. Melo, F. Pereira, and E. Costa, "Extended experiments with ant colony optimization with heterogeneous ants for large dynamic traveling salesperson problems," in *14th International Conference on Computational Science and Its Applications (ICCSA)*, pp. 171–175, 2014.

[29] U. Boryczka and Ł. Strąk, "Heterogeneous DPSO algorithm for DTSP,". in *Computational Collective Intelligence*, Springer, pp. 119–128, 2015.

[30] M. R. Khouadjia, E. Alba, L. Jourdan, and E. G. Talbi, "Multi-Swarm optimization for dynamic combinatorial problems: A case study on dynamic vehicle routing problem," in *ANTS Conference*, Springer, pp. 227–238, 2010.

[31] M. Okulewicz and J. Mańdziuk, "Two-phase multi-swarm PSO and the dynamic vehicle routing problem," in *IEEE Symposium on Computational Intelligence for Human-like Intelligence (CIHLI)*, pp. 1–8, 2014.

[32] R. Lung and D. Dumitrescu, "A collaborative model for tracking optima in dynamic environments," in *IEEE Congress on Evolutionary Computation (CEC)*, pp. 564–567, 2007.

[33] X. Zheng and H. Liu, "A different topology multi-swarm PSO in dynamic environment," in *IEEE International Symposium on IT in Medicine Education (ITIME '09)*, Vol. 1, pp. 790–795, 2009.

[34] R. I. Lung and D. Dumitrescu, "Evolutionary swarm cooperative optimization in dynamic environments," *Natural Computing*, Vol. 9, pp. 83–94, 2010.

[35] J. Branke, T. Kaußler, C. Smidt, and H. Schmeck, "A multi-population approach to dynamic optimization problems," in *Evolutionary Design and Manufacture*, Springer, pp. 299–307, 2000.

[36] H. Cheng and S. Yang, "Multi-population genetic algorithms with immigrants scheme for dynamic shortest path routing problems in mobile Ad Hoc networks," in *European Conference on the Applications of Evolutionary Computation*, Springer, Berlin, Heidelberg, pp. 562–571, 2010.

[37] T. Blackwell, "Particle swarm optimization in dynamic environment," in *Evolutionary Computation in Dynamic and Uncertain Environments*, Springer, Berlin, Heidelberg, pp. 29–49, 2007.

[38] T. T. Nguyen and X. Yao, "Benchmarking and solving dynamic constrained problems," in *IEEE Congress on Evolutionary Computation*, pp. 690–697, 2009.

[39] F. O. DeFranca and F. J. VonZuben, "A dynamic artificial immune algorithm applied to challenging benchmarking problems," in *IEEE Congress on Evolutionary Computation*, Piscataway, NJ, USA, pp. 423–430, 2009.

[40] C. Li and S. Yang, "Fast multi-swarm optimization for dynamic optimization problems," in *Fourth International Conference on Natural Computation*, Vol. 7, pp. 624–628, 2008.

[41] M. Kamosi, A. Hashemi, and M. Meybodi, "A new particle swarm optimization algorithm for dynamic environments," *Swarm, Evolutionary, and Memetic Computing*, Vol. 129–138, 2010.

[42] I. G. Del Amo, D. A. Pelta, J. R. González, "Using heuristic rules to enhance a multiswarm PSO for dynamic environments," in *IEEE Congress on Evolutionary Computation (CEC)*, pp 1–8, 2010.

[43] M. Daas and M. Batouche, "Multi-bacterial foraging optimization for dynamic environments," in *6th International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, pp 237–242, 2014.

[44] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, M. Meybodi, and M. Akbarzadeh-Totonchi, "mNAFSA: a novel approach for optimization in dynamic environments with global changes," *Swarm and Evolutionary Computation*, Vol. 18, pp. 38–53, 2014.

[45] F. B. Ozsoydan and A. Baykasoglu, "A multi-population firefly algorithm for dynamic optimization problems," in *IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, pp. 1–7, 2015.

[46] C. Li and S. Yang, "A general framework of multipopulation methods with clustering in undetectable dynamic environments," *IEEE Transactions on Evolutionary Computation*, Vol. 16, pp. 556–577, 2012.

[47] G. Dun-wei and S. Xiao-yan, "Decision-making models for resource allocation in multi-population co-evolutionary genetic algorithms implemented on networks," *International Journal of Computer Science & Network Security*, Vol. 6, pp. 239–245, 2006.

[48] B. Xu, Y. Zhang, D. Gong, Y. Guo, and M. Rong, "Environment sensitivity-based cooperative co-evolutionary algorithms for dynamic multi-objective optimization," *IEEE/ACM Transactions on Computational Biology & Bioinformatics*, Vol. 15, No. 6, pp. 1877–1890, 2017.

[49] H. Richter, "Detecting change in dynamic fitness landscapes," in *IEEE Congress on Evolutionary Computation (CEC)*, pp. 1613–1620, 2009.

[50] A. Simões and E. Costa, "Evolutionary algorithms for dynamic environments: Prediction using linear regression and Markov chains," in *International Conference on Parallel Problem Solving from Nature*, Springer, Berlin, Heidelberg, pp. 306–315, 2008.

[51] T. T. Nguyen, "Continuous dynamic optimization using evolutionary algorithms," *Ph.D. dissertation*, School of Computer Science, University of Birmingham, January 2011.

[52] X. Zou, M. Wang, A. Zhou, and B. McKay, "Evolutionary optimization based on chaotic sequence in dynamic environments," in *IEEE International Conference on Networking, Sensing and Control*, Vol. 2, pp. 1364–1369, 2004.

[53] A. Carlisle and G. Dozier, "Adapting particle swarm optimization to dynamic environments," in *International Conference on Artificial Intelligence*, Vol. 1, pp. 429–434, 2000.

[54] H. G. Cobb, "An investigation into the use of Hyper mutation as an adaptive operator in genetic algorithms having continuous," *Time-Dependent Non-stationary Environments, Technical Report AIC-90-001*, Naval Research Laboratory, Washington, USA, 1990.

[55] R. W. Morrison, "Designing evolutionary algorithms for dynamic environments," Springer-Verlag, Berlin, 2004.

[56] R. Eberhart and J. A. Kennedy, "New optimizer using particle swarm theory," in *IEEE Proceedings of the Sixth International Symposium on Micro Machine and Human Science (MHS)*, pp. 39–43, 1995.

[57] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, Springer, pp. 760–766, 2011.

[58] S. Mirjalili, "The ant lion optimizer," *Advances in Engineering Software*, Vol. 83, pp. 80–98, 2015.

[59] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *IEEE World Congress on Computational Intelligence (CEC)*, pp. 69–73, 1998.

[60] T. Blackwell, J. Branke, and X. Li, "Particle swarms for dynamic optimization problems," in *Swarm Intelligence*, Springer, pp. 193–217, 2008.

[61] I. Rezazadeh, M. Meybodi, and A. Naebi, "Particle swarm optimization algorithm in dynamic environments: Adapting inertia weight and clustering particles," in *Fifth UKSim European Symposium on Computer Modeling and Simulation (EMS)*, pp 76–82, 2011.

[62] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, and M. R. Meybodi, "A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization," *Applied Soft Computing*, Vol. 13, No. 4, pp. 2144–2158, 2013.

[63] D. Yazdani, B. Nasiri, R. Azizi, A. Sepas-Moghaddam, and M. R. Meybodi, "Optimization in dynamic environments utilizing a novel method based on particle swarm optimization," *International Journal of Artificial Intelligence*, Vol. 11, No. A13, pp. 170–192, 2013.

[64] J. K. Kordestani, A. Rezvanian, and M. R. Meybodi, "CDEPSO: A bi-population hybrid approach for dynamic optimization problems," *Applied Intelligence*, Vol. 40, No. 4, pp. 682–694, 2014.

[65] W. Du and B. Li, "Multi-strategy ensemble particle swarm optimization for dynamic optimization," *Information Sciences*, Vol. 178, pp. 3096–3109, 2008.

[66] R. Mukherjee, G. R. Patra, R. Kundu, and S. Das, "Cluster-based differential evolution with crowding archive for niching in dynamic environments," *Information Sciences*, Vol. 267, pp. 58–82, 2014.

[67] A. Boulesnane and S. Meshoul, "WD2O: A novel wind driven dynamic optimization approach with effective change detection," *Applied Intelligence*, Vol. 47, pp. 488–504, 2017.

[68] J. Kazemi Kordestani, H. Abedi Firouzjaee and M. Meybodi, "An adaptive bi-flight cuckoo search with variable nests for continuous dynamic optimization problems," *Applied Intelligence*, Vol. 48, pp. 97–117, 2018.

[69] M. Mohammadpour, H. Parvin and M. Sina, "Chaotic genetic algorithm based on explicit memory with a new strategy for updating and retrieval of memory in dynamic environments," *AI and Data Mining*, Vol. 6, pp. 191–205, 2018.

S. M. Ejabati received his B.Sc. and M.Sc. degrees in Electrical and Electronic Engineering from Guilan University in 2010 and Birjand University in 2013, respectively. In 2019, he completed his Ph.D. in Electrical and Electronics Engineering at University of Birjand, and his research interests include model recognition, dynamic optimization, and signal processing.

S. H. Zahiri received the B.Sc., M.Sc. and Ph.D. degrees in Electronics Engineering from Sharif University of Technology, Tehran, Tarbiat Modarres University, Tehran, and Mashhad Ferdowsi University, Mashhad, Iran, in 1993, 1995, and 2005, respectively. Currently, he is a Professor with the Department of Electronics Engineering, University of Birjand, Birjand, Iran. His research interests include pattern recognition, evolutionary algorithms, swarm intelligence algorithms, and soft computing.