Engineering: An International Journal

Journal Homepage: www.engijournal.com

A New Efficient Approach to Deal With Dynamic Optimization Problems

Michael Abraham

Abstract: Along with increasing scientific progress, humans are constantly confronted with several new real-world issues. This further demonstrates the need for optimization algorithms that can quickly adapt to an uncertain and changing environment over time. In such issues, the current conditions lead to an area of optima and worth changed after some time. Therefore, an optimization calculation must have the option to adjust rapidly to evolving conditions. This paper proposes a new algorithm dependent on the PSO calculation, alluded to as a versatile increasing/decreasing PSO calculation. In the optimization procedure, this algorithm can generally adaptively find and track the ideal number changed after some time in nonlinear and dynamic conditions with imperceptible changes, by decreasing or expanding the quantity of calculation particles and the successful hunt extend. Also, another definition has been presented called the Focused Search Zone, which expects to feature promising spaces to quicken the neighborhood search process and forestall untimely combination and achievement file as a paradigm for deciding centered pursuit zone conduct toward ecological conditions. The consequences of the proposed calculation are assessed on the moving pinnacle benchmark work and hence contrasted and those got from a few legitimate calculations. The outcomes show a beneficial outcome of versatile systems utilized, remembering a reduction or an expansion for particles and search extend on finding and following numerous optima contrasted with other multi-populace based streamlining calculations.

Keywords: Adaptive Search Radius, Dynamic Optimization Problems, Local Search, Multi-Search, Particle Swarm Optimization Algorithm

Paper history: Received 13 April 2020; Received in revised form 25 April 2019; Accepted 1 May 2020; Available online 10 May 2020.

1 INTRODUCTION

REAL-WORLD issues, such as those that occur in nature, are constantly changing over time. Therefore, nowadays, a spotlight has been shown on the use of nature-inspired evolutionary and collective intelligence algorithms to solve dynamic optimization problems (DOPs) [1, 2]. Unlike static environments, the location of optima is constantly changing as time goes on in dynamic environments. Hence, optimization algorithms not only have to find a globally optimal solution in a particular search space but also have to constantly track global changes in optimum and sometimes several optima close to it in different environments with nonlinear changes. Thus, algorithms must be able to adapt to changes in the environment.

Consequently, there is a major difference between dynamic and static environments in terms of goals, challenges, performance measurement, and optimization benchmark functions.

Challenges facing static environments include premature convergence, get stuck in a local optimum, and reconciliation between exploration and extraction. Dynamic environments, on the other hand, face several other challenges in addition to the above. These include detecting changes in the environment, converting local optimum to the global optimum and vice versa, losing diversity after a change, obsolete memory after a change, more than an optimum surrounded by a sub-population unaware of the extent and manner of changes in the environment, and changes in the part of the environment unaware of the time of change in the environment.

In recent years, several different methods have been proposed to promote traditional evolutionary algorithms that can be used in static environments. They can be divided into 8 groups:

-Increased diversity after changes in the environment [3-9]

- Diversity preserved during execution [9-14]

- Memory allocation schemes [15 and 16]

• **Corresponding author**: *Michael Abraham is with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University. E-mail: michael.abraham@asu.edu*

- Multi-population methods [17-23]
- Hybridization [24-26]

According to many studies on solving DOPs, locating and tracking a set of optima is more effective than locating and tracking a global optimum [27-29]. This is because tracking a global optimum in a set of the best optima is more efficient when the environment changes over time. In this regard, the application of multi-population methods seems to be more efficient than others in locating and tracking several local optima close to global optima in dynamic problem optimization. In this method, the entire search space is divided into a series of sub-spaces. Each sub-space is known as a sub-population that covers one or more of the local optima. From now on, the algorithm updates the particles of each sub-population separately and searches for a better optimum. In multi-population methods, the process of creating an acceptable number of sub-populations and individuals to cover different sub-spaces in the search space is considered a challenging issue. For example, in their research, Lee Young used a hierarchical clustering method to automatically divide the search space into sub-populations [29, 30].

Another problem faced by dynamic problem optimization is determining the time of change in the environment by the algorithm. Many dynamic optimization algorithms have been developed to detect changes in the environment. Such algorithms begin to disperse particles and increase diversity in the new environment as soon as they feel a change in the environment. Suppose a mode in which only part of the entire search space changes; then, it becomes very difficult to predict the sub-space or to detect changes, and all algorithm functions are overshadowed. Hence, algorithms that do not need to detect changes in the environment perform better in different environments.

This paper proposes an Adaptive Increasing/decreasing Particle Swarm Optimization (AidPSO) algorithm for effective use of the multi-population method in nonlinear and time-variable environments. The introduced algorithm is based on the PSO algorithm for dynamic environments. Several new solutions have been proposed to solve problems such as creating sub-populations, preventing premature convergence, a swarm of individuals and their numbers, and adapting people's behavior to environmental conditions.

The moving peaks problem, one of the most well-known optimization problems in dynamic environments, has been employed to evaluate the proposed algorithm [31 and 32]. To evaluate it, a comparison has been made between AidPSO and several other algorithms that utilize multi-population methods to solve DOPs. Furthermore, a series of experiments have been performed to investigate the adaptive mechanisms that control the behavior of the algorithm under different environmental conditions.

The remainder of this paper is structured as follows. Section II reviews developed multi-population methods for solving dynamic problems. Section III describes the structure of the proposed AidPSO algorithm. Section IV presents the performance of the algorithm in coping with the moving peak benchmark function with different settings. Finally, Section V provides a conclusion and discussion.

2 Optimization Using Multi-Population-Based Methods

This section reviews several proposed algorithms in this field.

The clustering PSO (CPSO) algorithm is presented in [29 and 30]. It uses a hierarchical clustering method to divide the initial population into several sub-populations to cover different local zones.

A multi-population PSO algorithm based on the island model is introduced in [33]. In this algorithm, particles migrate regularly between different populations. In [24], the algorithm performs differently, where a connection is established between populations only when a change is observed in the environment.

In [34], the entire population is divided into two categories, the parent population, which is tasked with searching the search space, and the child population, which is tasked with tracking local optima. The parent population is constantly studying the situation to create a child population.

It is worth noting that the total number of people is always a fixed amount. The Fast-Multi-Swarm Optimization Algorithm (FMSO) [35] was introduced in which the parent population, as the base population, searches for more feasible zones after changes in the environment and child population is used to search locally in these feasible zones, as the only difference. A similar idea has been proposed in [36] in which the child population, if ineffective, becomes hibernated until a change in the environment is felt.

Blackwell and Brank have used the properties of atoms and the principle of repulsion between particles of the same charge to maintain diversity in populations [37]. Charged populations are introduced to cover optima separately. Two innovative rules have been added to its improved version [38] to increase diversity. According to one rule, the number of quantum particles increases and the number of particle pathways decreases as the environment changes. According to the second rule, particles that have an unacceptable performance are either initialized or stopped.

Additionally, much of the research that has been done so far [39] on solving dynamic optimization problems based on evolutionary algorithms is based on detecting changes in the environment [29, 30, and 37] or predicting changes assuming that a specific pattern is being followed [40]. Upon the detection or anticipation of change, a variety of strategies are used to increase diversity.

3 PROPOSED ALGORITHM

The proposed algorithm divides the particles into two groups, i.e., free particles and focus particles, to synchronize conventional PSOs for dynamic environments.

Free particles are required to search permanently throughout the search space. As these particles tend to converge, several located in the so-called Focused Search Zone (FSZ), are introduced as focus particles. The FSZ range around the best focus particle is determined by a radius called Adaptive Search Radius (ASR). Following the update of focus particles, if the best particle is recovered, the FSZ center will be transferred to the location of the best focus particle and the other focus particles of this FSZ will be forced to deploy and search in this sub-space. At this time, all free particles are dispersed in the search space to locate other optima. Free particles are required to do an initial search to find feasible sub-spaces. Focus particles, on the other hand, are required to find and improve the optimum and track it while applying changes to the environment. The number of focus particles and the value of ASR is controlled by a metric called the success index. The number of particles is constantly changing adaptively.

Given the importance of maintained diversity in solving DOPs and their dependence on detecting environmental change, the performance of proposed algorithms for dynamic environments should not depend on identifying changes in the environment. The proposed algorithm does not need to detect changes in the environment and always adapts itself to environmental conditions.

3.1 Adaptive Increasing/Decreasing PSO Algorithm

This section introduces a new algorithm called "adaptive increasing/decreasing PSO algorithm (AidPSO)" to optimize in dynamic environments (time-variable). Thanks to its short mechanism and rapid adaptation to environmental conditions, this algorithm has acceptable performance in solving DOPs with high-speed changes. Algorithm 1 shows the general framework of the algorithm.

The main loop of the proposed algorithm begins with the initialization of free particles, followed by updating the locations of these particles with conventional PSO. Afterward, the convergence of free particles is examined. If convergence is likely to be detected, the particles in the FSZ, i.e., in the ASR range, around the best particle, will be recognized as focus particles, leading to a new FSZ. Now, the focus particles located in each FSZ are updated separately by the PSO; if the optimum located in each FSZ is improved, the FSZ center is transferred to a more suitable detected optimum. FSZs are then examined for superimposition (superposition) after a specific time called “evolution time,” laying the groundwork for ASR correction. Next, an upgrade time is allocated to the FSZs that have lost their optimum as a result of drastic changes in environmental conditions, to recover their optimum so that they are not deleted immediately. This upgrade time is allocated because the optimum is transferred to the zones around its previous location following a change in the environment, in most cases.

In this case, following the allocation of upgrade time and an increase in ASR and the number of focus particles, the optimum in question can be traced more quickly. Finally, ASR and the number of focus/free particles are calculated for the current environmental conditions using the success index.

Algorithm 1. The general framework of the proposed algorithm

Algorithm 1 Aid*PSO*

```
1   Initialize the free particles
2   while stop criteria is not satisfied
3       for free particles do
4           PSO();
5       end for
6       convergenceChecking (free particles)
7       if FSA exist 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      ASRoverlapChecking (FSZ,ASR)
15      RemoveUselessFSZ
16      CalcASR&NumOfFocusPartc
17      CalcNumOfFreePartc
18                  end while
```

The proposed algorithm divides the particles into two groups: free and focus. In the first step, a predetermined number of free particles is initialized. In the second step, their number is always determined by the algorithm. One of

the features of the proposed algorithm is the number of focus/free particles variable over time, depending on environmental conditions. Thus, during the execution of the algorithm, the total number of particles is always subject to environmental conditions. Furthermore, the number of particles decreases proportionally when the algorithm converges toward the optimum. Thanks to this feature, the algorithm is separated from the constant update of a large number of particles. In the third step, the position of free particles is updated through the PSO algorithm, proposed for the first time by Kennedy and Eberhart. Algorithm 2 shows the PSO algorithm implementation process.

Algorithm 2. PSO Algorithm

Algorithm 2 *PSO*

```
1   Evaluate the fitness of each particle
2   for each particle i do
3       Update particle i according to (1) and (2);
4       If f(x_i) < f(x_pbest_i)  then
5           x_pbest_i := x_i;
6           If f(x_i) < f(x_gbest_i)  then
7               x_gbest_i := x_i;
8           end if
9       end if
10  end for
```

Each particle i with a velocity vector $\vec{v}_i$ and a position vector $\vec{x}_i$, is represented, updated by version [42] with inertia weight as follows:

$$v'^{d}_i = \omega v^d_i + c_1 r_1 \left(x^d_{pbest_i} - x^d_i\right) + c_2 r_2 (x^d_{gbest} - x^d_i) \quad (1)$$
$$x'^{d}_i = x^d_i + v'^{d}_i \quad (2)$$

In the above equation, x'^{d}_i is the current position, x^d_i is the previous position, v'^{d}_i is the current velocity, and v^d_i is the previous velocity of the ith particle at the dth dimension. $x^d_{pbest_i}$ is the best position of the ith particle so far and x^d_{gbest} is the best position among all the particles. $\omega \in (0,1)$ is the inertia weight that determines the degree to which the previous velocity affects the current velocity of the particle. r_1 and r_2 are random numbers between 0 and 1. c_1 and c_2 are called acceleration constants, indicating the degree to which the particle follows its best (the cognitive component) and collective best (social component), respectively.

Then, the main loop of the proposed algorithm, i.e., the free particle convergence process, is examined, shown in algorithm 3. The initial convergence condition is met by measuring the changes in the position of x_{gbest} free particles and their fitness compared to the previous step. If the changes in x_{gbest} are less than $r_{conv}/5$ and the changes in its fitness are less than r_{conv}, the algorithm will detect the possibility of free particle convergence. At this stage, the position of the best located point is examined. If the point is located in any of the FSZs, with its fitness value better than that of the FSZ center in it, the FSZ and subsequent focus particles are removed, leading to the formation of a new FSZ with free particles located in the ASR range with x_{gbest} at the center. Besides, all free particles are dispersed to search for other feasible zones throughout the search space. On the other hand, all free particles will disperse if the x_{gbest} fitness value is worse than the FSZ center in it.

Following a satisfied initial convergence condition, if the best point of free particles is not located in any of the FSZs, the total distance between the two free particles with the shortest distance from x_{gbest} will be explored. A sum of less than r_{conv} indicates that the free particles are converging, and x_{gbest} is considered as the center of FSZ. Focus/free particles located in the ASR range relative to this center are introduced as the focus particles of that FSZ. All free particles are dispersed to search for other feasible zones throughout the search space. Moreover, an "evolution time," during which the category is provided with the opportunity to improve its optimum, is considered so that this category is not deleted by the algorithm in the next steps as an additional category.

In the next step, the position of the focus particles of each FSZ is updated separately by the PSO algorithm followed by transferring the center of that FSZ to the position of the best particle if the best particle of each FSZ is improved. Thus, the location of FSZs in the search space is also updated. The algorithm also monitors the velocity of focus particles to prevent them from getting caught up in local optimum by dispersing them within their FSZ range whenever they get close to zero.

The proposed algorithm uses a special mechanism to prevent the overlapping of FSZs and focus particle parallelism. In this method, the distances between FSZ centers are calculated. Then, two neighboring FSZs with a center distance of less than twice the ASR is considered. These categories must have reached the so-called relative stability, meaning that they must have spent the evolution time of the new FSZs and the upgrade time of the FSZs that have lost their optimum under the new environmental conditions. Now, to avoid the algorithm being influenced by local optima, the mean fitness of optima detected by other FSZs is calculated. If the fitness of the two FSZ centers is better than a factor (

(P_{mean}) of the mean fitness of the centers of other FSZs, the search radius is reduced by half the distance between the centers of these two FSZs so that both can encompass and track their optimum.

Algorithm 3. Checking free particle convergence

```
Algorithm 3 convergenceChecking ()
If (f(xgbest)t_1 - f(xgbest)t < rconv) and (|| (xgbest)^t, || < rconv/5) then
   If xgbest was within FSZ[i] then
      if f(xgbest) better than f(centerFSZ[i]) then
        Replace selected free particles with focus particles in FSZ[i]
         re-initialize the free particles
      else
       re-initialize the free particles
      end if
   else
      create a new FSZ with selected free particles as the focus particles
      re-initialize the free particles
   end if
                                   end if
```

Now, if the fitness of one or both FSZs is worse than a factor (P_{mean}) of the mean fitness of the centers of other FSZs, the FSZ with a worse detected optimum fitness with focus particles compared to the other, will be considered as the local optimum and will be removed along with focus particles linked to it. Thus, searching for multiple FSZs in a zone of the search space and getting caught up in local optima are both prevented, increasing the speed of the algorithm to find more optima in the entire search space in less time. Algorithm 4 shows the process of checking the overlapping of FSZs.

Algorithm 4. Checking FSZ Overlapping

```
Algorithm 4 ASRoverlapChecking (FSZ,ASR)
if FSZ[i] & FSZ[j] are stable and distance between FSZ[i] center and FSZ[j] center < 2*ASR then
   fmean(centerFSZ) = Calculation of mean fitness other centers
   if f(centerFSZ[i]) and f(centerFSZ[j]) beter than Pmean* fmean(centerFSZ) and ASR>1 then
  ASR = (distance between FSZ[i] center and FSZ[j] center) / 2
     else
       delete FSZ with worst fitness between FSZ[i] and FSZ[j]
     end if
  end if
```

Another idea used in the proposed algorithm shown in algorithm 5 is to allocate upgrade time to FSZs that no longer see the optimal point in their zone or their focus particles are stuck in the local optimum, following drastic changes in the environment. Upgrade time allows these FSZs to locate and track their optimum. Furthermore, focus particles stuck in local optimum are provided with the opportunity to be released with an increase in speed during the upgrade. Here, too, a factor (P_{mean}) of the mean fitness of detected optima is introduced as a measurement criterion. At each stage, the corresponding FSZs are removed along with their focus particles after the upgrade time, following the comparison of the worst optimum in FSZs with the metric above if they have a more unacceptable value.

Algorithm 5. Removal of unnecessary FSZs

```
Algorithm 5 RemoveUselessFSZ
   fmean(centerFSZ) = Calculation of mean all fitness centers
   if FSZ[i] is stable and worst f(centerFSZ[i]) > fmean(centerFSZ) then
      delete FSZ with worst fitness FSZ[i]
      End if
```

One of the outstanding features of this algorithm, which has led to its increased speed in finding the optimum, is the observation of environmental conditions and subsequently adjusting the behavior of the algorithm by managing the number of particles and the search range.

Thus, the success index is defined as a criterion for determining the degree of optimization of the algorithm. Regarding this index, a criterion is obtained for the overall behavior of the categories when faced with environmental conditions according to the optimal mean of the categories by comparing it with the worst optimal value detected (Equation 3).

$$success_{Index} = \frac{mean\_fit - wrost\_fit}{mean\_fit} \quad (3)$$

Following the determination of a metric for the behavior of the algorithm and environmental conditions, the search radius and the number of focus particles of the categories should be adapted to the environmental conditions to optimize the performance of the algorithm as much as possible. If changes are made to the environment, some FSZs will lose their optimum and subsequently, the success rate will decrease. As previously mentioned, in this case, an upgrade time is allocated for these FSZs to be provided with the opportunity to adapt to new environmental conditions and thus detect the optimum under their coverage before being removed by the algorithm as an unnecessary category. To speed up the optimization process according to the value of the success index, the algorithm proceeds to increase the ASR and the number of focus particles (Num_FP) in FSZ according to the following equations because it is unaware of the degree of changes applied in the environment.

$$ASR^{t+1} = ASR^t + success\_Index * ASR^t \quad (4)$$
$$Num\_FP^{t+1} = Num\_FP^t + success\_Index * Num\_FP^t \quad (5)$$

The detected optimum is improved during FSZ upgrades, manifested in an increased success index. In this case, the ASR decreases according to Equation (6) to obtain a more focused search.

$$ASR^{t+1} = ASR^t - \frac{Success\_Index}{1+Success\_Index} * ASR^t \quad (6)$$

Finally, if there is no FSZ in the upgrade period, ARS and the number of focus particles (Num_FP) are reduced to achieve a more focused search and increased algorithm speed, according to an improved success index based in the following equations.

$$\Delta Success\_Index = Success\_Index^{t-1} - Success\_Index^t \quad (7)$$
$$ASR^{t+1} = ASR^t + \Delta Success\_Index * ASR^t$$
$$Num\_FP^{t+1} = Num\_FP^t - Success\_Index * Num\_FP^t \quad (8)$$

To increase the speed of the algorithm, the number of free particles is determined, always based on the number of real optima found. The number of optima is measured when all categories have passed their upgrade period. In this case, the measurement criterion is the number of FSZs that are not in the evolution period, reached so-called relative stability. According to Equation (9), the number of free particles (Num_FreeP) is calculated to find new feasible zones based on the number of optima (Num_optm) detected.

$$Num\_FreeP = 15 + 2\log(Num\_optm) \quad (9)$$

4 Simulation Results

This section employs the proposed algorithm to solve the MPB problem [31]. First, the MPB problem and the measurement procedure of the performance of dynamic algorithms are introduced. Then, the experiments are divided into two parts. The purpose of the first part of the experiments is to investigate the working mechanism of AidPSO adaptive parameters when dealing with the MPB problem. The purpose of the second part is to compare AidPSO with several algorithms introduced in this field in terms of performance. The results of all the algorithms introduced in this paper are derived from the results and suggestions of previous studies.

In the PSO algorithm, the acceleration constants c_1 and c_2 indicate the degree to which the particle follows its best (the cognitive component) and collective best (the social component), respectively, and changes the inertia weight convergence rate. Due to the different tasks considered for focus/free particles in AidPSO, different acceleration and inertia weight constants have been considered to update free particles, tasked with finding feasible zones in the entire search space and focus particles, tasked with locating and tracking the optimum in their FSZ. The values of c_1, c_2, and ω are considered to be 2, 2, and 0.4 for free particles with a general search and 3, 1, and 0.5 for focus particles with a detailed search, respectively.

4.1 Moving Peak Benchmark (MPB) Problem

The MPB function [31] is one of the most well-known optimization problems in dynamic environments, widely used to evaluate dynamic optimization algorithms. In the MPB problem, the optimum can be changed by three features: position, height, and width of the peaks. This problem is defined in *D* dimensions as follows:

$$\boldsymbol{F(\vec{x}, t)} = \max_{i=1,2,\ldots,p} \frac{H_i t}{1+W_i(t)\sum_{j=1}^{D}(x_j(t)-x_{ij}(t))^2} \quad (10)$$

In Equation (10), $H_i(t)$and $W_i(t)$ are the height and width of the peak i at time t, respectively, and $x_{ij}(t)$ is the jth element of the peak location i at time t. The parameter p is mixed through the max function independently for certain peaks. Peak displaces along a random direction through vector $\vec{v}_j$ as much as a distance s, which represents the sensitivity of the problem dynamics. The movement of a lone peak is defined by the following equation:

$$\vec{v}_j(t) = \frac{s}{|\vec{r}+\vec{v}_i(t-1)|}\left((1-\lambda)\vec{r} + \lambda\vec{v}_i(t-1)\right) \quad (11)$$

The transfer vector $\vec{v}_j(t)$ is a linear combination of the random vector $\vec{r}$ and the previous transfer vector $\vec{v}_i(t-1)$, normalized relative to the shift length s. The value of the correlation parameter λ is considered to be zero, indicating the inconsistency of peak movements. The equations of changes in a peak are expressed as follows:

$$H_i(t) = H_i(t-1) + height\_severity \times \sigma \quad (12)$$
$$W_i(t) = W_i(t-1) + width\_severity \times \sigma \quad (13)$$
$$\vec{X}_i(t) = \vec{X}_i(t)(t-1) + \vec{v}_i(t) \quad (14)$$

where σ is a random number with a normal distribution, mean of 0, and variance of 1.

As with other studies, the default settings of the benchmark function are shown in this paper in Table 1. The term "change frequency, U" means the change in the environment after U times of fitness function evaluation. The peak location range is the search space range in each dimension. The height of the peak changes randomly in the range [30, 70] and its width in the range [1, 12].

Table 1. Default settings for moving peaks

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

4.2 Algorithm Performance Measurement

Numerous measurement criteria have been introduced to measure the performance of algorithms in dynamic environments [43]. To yield results comparable to those of other algorithms in this field, the offline error (OE) criterion, defined as the mean difference between the optimum value found by the algorithm and the global optimum value in each environment, has been used.

$$OE = \frac{1}{K}\sum_{k=1}^{K}(h_k - f_k) \quad (15)$$

In the above equation, f_k is the best solution found by the algorithm before the kth change in the environment and h_k is the optimum value of the kth environment. OE is the average difference between f_k and h_k in the total number of K changes in the environment. All reported results are related to more than 50 times the program running for 100 changes in the environment.

4.3 Investigating the Adaptive Parameters of the Algorithm

A series of experiments have been conducted on MPB with default values to demonstrate the algorithm's decision-making process for ASR values and the number of focus/free particles under different environmental conditions.

As previously mentioned, a higher percentage of optimizations change their location around the previous location when applying changes to the environment.

Therefore, optima can be detected with higher speed and accuracy following changes in the environment with a simultaneous increase in the search range and number of particles in each FSZ. Figure 1 shows the performance of the algorithm in setting ASR values and the number of focus particles located in each FSZ when faced with changes in en-

vironmental conditions. Obviously, when the environment changes, the algorithm attempts to increase the radius and number of focus particles in each zone according to the degree of decrease in the success index. Subsequently, the radius and number of focus particles decrease following an improved success index for an increased algorithm speed and focus on finding new optima.

To visualize the behavior of the algorithm before and after the change in the environment, Figure 2 is presented for a two-dimensional environment. In this figure, free particles are marked with blue + signs, peak location with black squares, focus particles with blue dots, FSZ with blue circles along with its centers with red stars.

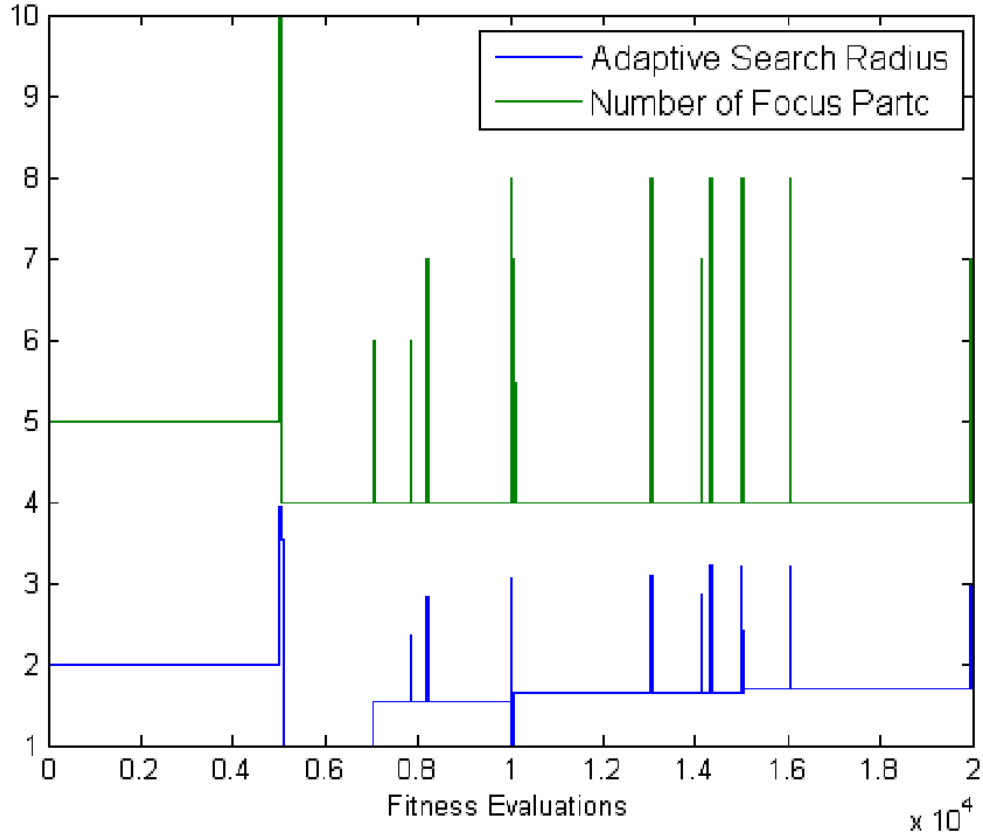

Fig. 1. Algorithm performance in setting the number of focus particles in each FSZ and ASR.

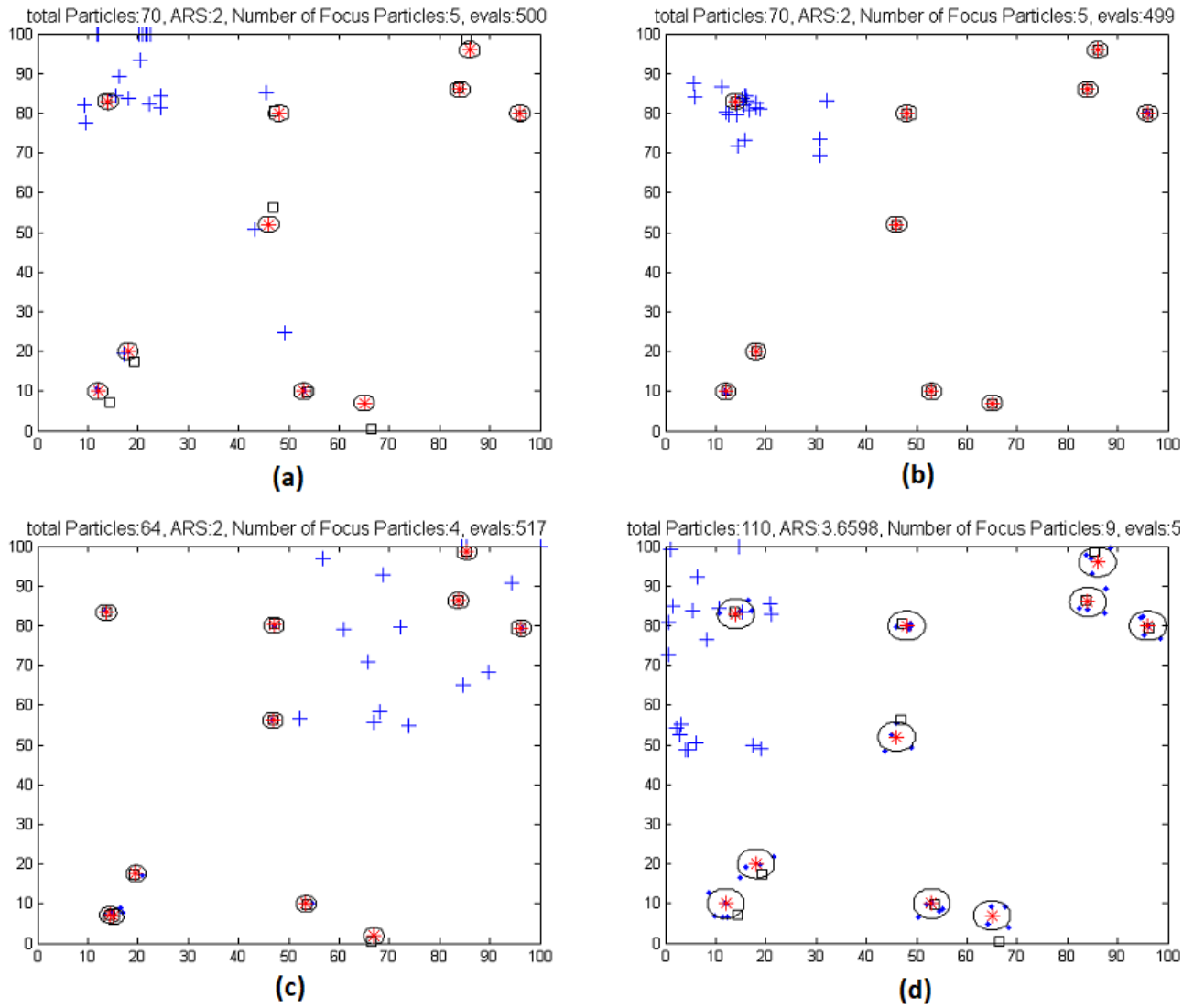

Fig. 2. Algorithm process in finding peaks and changes in the number of particles and ASR before and after the change in the two-dimensional environment. a) an iteration before a change in the environment, b) a change in the environment, c) an iteration after a change in the environment, d) 17 iterations after a change in the environment.

The optimization process from (a) to (d) suggests that the algorithm gets started with free particles. Upon finding the first feasible zone in the seventh iteration of fitness assessment and creating the first FSZ, 5 particles with better fitness are recognized as focus particles. Subsequently, all free particles are dispersed, and the same process continues to find all the peaks.

Figure 2 (a) shows a step before the change in the environment in which all peaks are surrounded by FSZs. After a change in the environment, the peak location changes and it is decided to increase the search radius and the number of focus particles in each FSZ following a decrease in the success index of the algorithm. According to Figure 2 (b), most peaks are located in FSZ after a change in the environment by this mechanism and a temporary increase in the number of focus particles leads to an increase in the velocity of the corresponding peaks. In Figure 2 (c), ASR increases to 3.66 and the number of focus particles to 3 for each FSZ. Following a relative enhancement in the success index, ASR values and the number of focus particles decrease to 2 and 4, respectively. Ultimately, the values of peaks in none of the FSZs are found by the free particles and the FSZs that contain no peaks are removed by the algorithm. The search radius and number of focus particles decrease following an improvement in the success index (Figure 2d).

4.4 Comparison with Other Algorithms

This part of the experimentation compares the proposed algorithm with other algorithms introduced in this topic, including CPSO [29, 30], mCPSO [37], mQSO [37], CESO [43], rSPSO [44], SPSO [28], AmQSO [45], mPSO [46], APSO [47], FTMPSO [48], SFA [49], PSO-AQ [50], CDEPSO [51], and CbDE-wCA [52], in solving the MPB problem with different settings. Table 2 shows the optimum values of AidPSO parameters to solve the MPB problem with the default values.

Table 2: Initial AidPSO settings to solve the MPB problem

Parameter	Value
rconv	0.5
Pmean	0.3
Evolution time	20 iteration
Upgrade time	30 iteration

The shift length and change frequency in the U-environment on the OE of the proposed algorithm, are offered following the experiments performed to depict the effects of the change in the number of peaks.

The set of experiments gathered in Table 3 shows how the algorithm deals with a change in the number of peaks for the MPB problem. A comparison has been made between the proposed algorithm and 14 additional algorithms in terms of OE and standard deviation (SD). The values provided for the other algorithms are derived from the papers by previous researchers with optimal settings. According to Table 3, the proposed algorithm has yielded better results for the total number of peaks compared to other algorithms in terms of solving the MPB problem.

From a closer look at the OE values of AidPSO and other algorithms in Table 3, it can be stated that the value of OE decreases with an increase in the number of peaks. This is because an increase in the number of detected local optima leads to an increase in the probability that the value of the fitness of these optima is closer to that of the global optima. Furthermore, more zones are monitored and searched in the search space by focus particles. As a result, the likelihood of the presence of global optima in these areas increases further following a change in the environment.

Table 4 presents the OE values for the 6 similar algorithms as well as the AidPSO algorithm for four different shift lengths. Environmental settings are the same default values and shift length values of 1, 2, 3, and 5.

Table 3. Comparison of OE of algorithms for a different number of peaks in the MPB problem in the change frequency range 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		1.04(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	1.42(0.06)	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	2.64(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	2.62(0.10)	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	2.42(0.05)	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	0.53(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	0.18(0.01)	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	0.42(0.03)	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	0.34(0.02)	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	0.41(0.00)	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)
AidPSO	0.14(0.03)	0.02(0.01)e-1	0.32(0.05)	0.39(0.03)	0.37(0.04)	0.17(0.01)	0.15(0.02)	0.13(0.04)
	0.15(0.02)e-10							

As it turns out, the peak location shift increases following a change in the environment with an increase in shift length. In other words, as the shift length increases further, the location of optimum shifts to a farther distance after the change in the environment, leading to more difficult optimum tracking for the algorithm. Thus, according to Table 4, the OE values for the algorithms increase. An incremental limited OE for the increased shift length for the proposed algorithm indicates the high stability of the algorithm in locating and tracking the optimum under any circumstances.

Table 4. Comparison of OE of algorithms for different shift lengths 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)
rSPSO	1.50(0.08)	1.87(0.05)	2.40(0.08)	3.25(0.09)
SPSO	2.51(0.09)	3.78(0.09)	4.96(0.12)	6.76(0.15)
AidPSO	0.32(0.05)	0.41(0.07)	0.46(0.06)	0.49(0.09)

Table 5. Comparison of OE of algorithms for a different number of peaks in the MPB problem at a change frequency of U = 500

Algorithm	**Number of peaks, *p***							
	1	5	10	20	30	50	100	200
mQSO(5,5q)	33.67(3.42)	11.91(0.76)	9.62(0.34)	9.07(0.25)	8.80(0.21)	8.72(0.20)	8.54(0.16)	8.19(0.17)
AmQSO	3.02(0.32)	5.77(0.56)	5.37(0.42)	6.82(0.34)	7.10(0.39)	7.75(0.32)	7.34(0.31)	7.48(0.19)
mPSO	8.71(0.48)	6.69(0.26)	7.19(0.23)	8.01(0.19)	8.43(0.17)	8.76(0.18)	8.91(0.17)	8.88(0.14)
APSO	4.81(0.14)	4.95(0.11)	5.16(0.11)	5.81(0.08)	6.03(0.07)	5.95(0.06)	6.08(0.06)	6.20(0.04)
FTMPSO	1.76(0.09)	2.93(0.18)	3.91(0.19)	4.83(0.19)	5.05(0.21)	4.95(0.15)	5.31(0.11)	5.52(0.21)
SFA	4.72(0.12)	4.88(0.12)	5.11(0.14)	5.72(0.13)	5.97(0.12)	5.94(0.15)	6.15(0.08)	6.18(0.11)
AidPSO	0.35(0.04)	2.29(0.03)	2.53(0.04)	1.89(0.03)	0.81(0.03)	1.45(0.01)	1.11(0.02)	1.03(0.02)

The changed frequency of the "change in the U environment" determines the time spent by the algorithm to locate the optimum

in each environment before any change. Obviously, a lower frequency means a much shorter time to detect the optimum. Tables 5-7 present the results of the MPB problem solution or a different number of peaks and environmental change frequencies of 500, 1000, and 10000. For example, an OE of 2.53 was obtained for 10 peaks at a change frequency of 500. Following an increase in frequency up to 1000, the algorithm is provided with more time and OE reaches 1.18. At a change frequency of 10,000, the OE decreases to 0.03.

Table 6. Comparison of OE of algorithms for a different number of peaks in the MPB problem at a change frequency of U = 100

pal	**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	0.70(0.10)	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)
AidPSO	0.1(0.05)e-13	0.04(0.00)	0.03(0.00)	0.04(0.00)	0.04(0.00)	0.01(0.00)	0.52(0.02)e-2	0.58(0.03)e-2

Table 7. Comparison of OE of algorithms for a different number of peaks in the MPB problem at a change frequency of U= 10000

Algorithm	Number of peaks, p							
	1	5	10	20	30	50	100	200
mQSO(5,5q)	18.60(1.63)	6.56(0.35)	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)
AidPSO	0.01(0.00)	1.23(0.04)	1.18(0.05)	1.03(0.06)	0.81(0.03)	0.76(0.03)	0.72(0.03)	0.75(0.05)

Figure 3 shows the "fitness value" graph based on the number of calls, compared to other methods. An environmental change frequency of 1000 is intended to demonstrate the ability and speed at which the algorithm adapts to changes in environmental conditions. Other MPB problem settings are done by default. Compared to other algorithms, particles converge to higher values at higher speeds in the proposed algorithm.

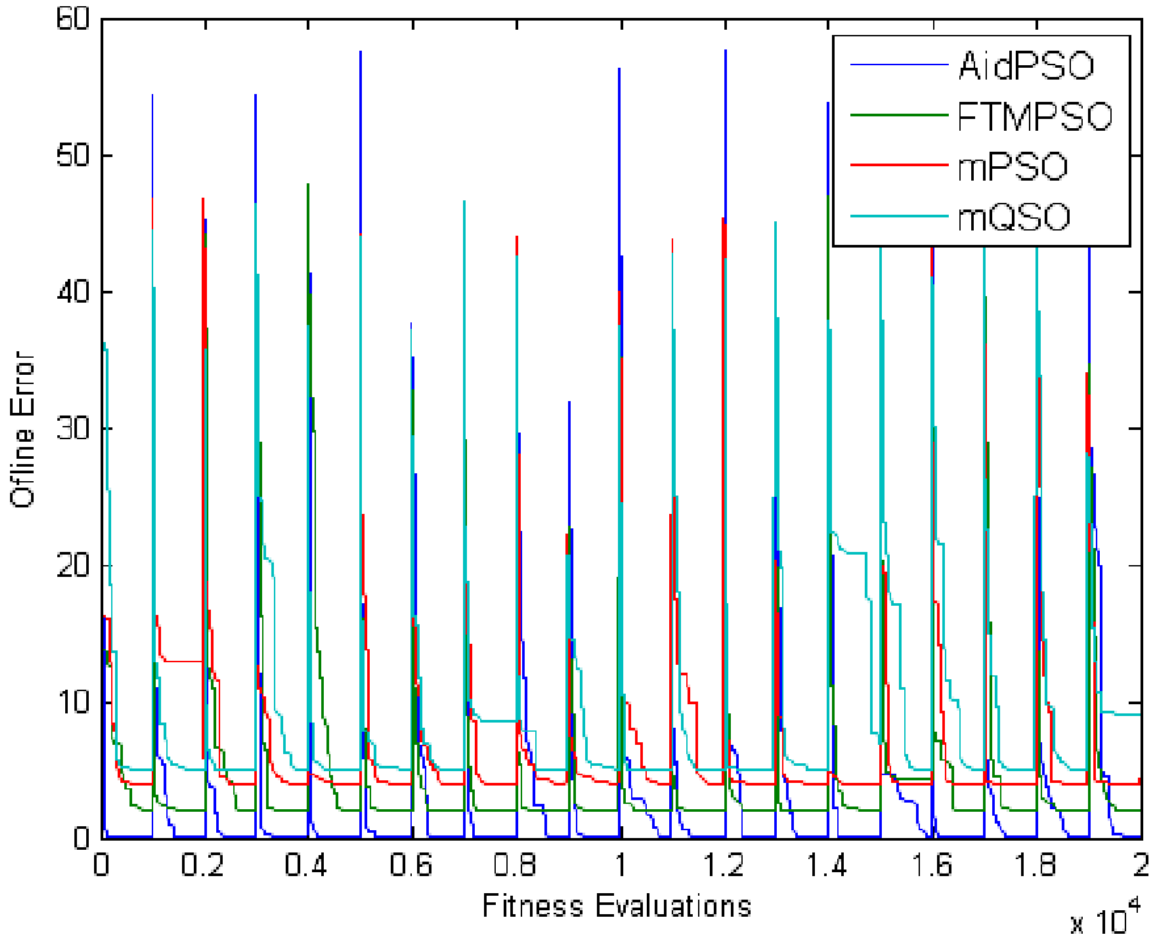

Fig. 3. Comparing OE with 10 peaks and 1000 frequency.

Table 8 shows the execution speed of the proposed algorithm. The data are presented for a one-time change in a five-dimensional environment with 10 peaks and a frequency of 1000 for the AidPSO algorithm and five other algorithms. The results of this table and those of Table 6 indicate a rapid and accurate performance of the algorithm in achieving a more optimal solution. The algorithm has achieved a better OE compared to other algorithms in less time.

Table 8. Comparison of algorithm run time for MPB problem

Algorithm	Time, s
CPSO	79.2394
mCPSO	81.8219
mQSO(5,5q)	80.3029
CESO	81.2834
rSPSO	78.3975
AidPSO	65.8952

5 Conclusion

These days, we are witnessing the widespread use of evolutionary algorithms to solve optimization problems in optimal dynamic environments that change with time, thanks to the nature-inspired mechanisms of action and the evolving natural conditions over time. In this regard, it is important not only to detect the optimum but also to track the detected optimum. To achieve this goal, many studies have suggested the use of multi-population methods.

This paper proposed a new algorithm for optimization in dynamic nonlinear environments. The adaptive increasing/decreasing PSO (AidPSO) algorithm increases or decreases the number of particles adaptively according to environmental conditions to locate and track the optimum. This algorithm divides particles into two groups, free particles, which are responsible for detecting feasible zones and focus particles, which are responsible for locating and tracking the optimum in feasible zones. The total number of particles and the size of the feasible zones are controlled with a higher probability of optimum detection by using several mechanisms.

In dynamic optimization, the reduction of optimization time is important, with a direct relationship with a faster convergence of the algorithm to the global optimum. In dynamic problems, this convergence must be accompanied by a diversity preserved throughout the search space. Hence, the majority of optimization algorithms used in dynamic problems get started with a large number of particles, followed by a decrease in the number of particles as the optimization process progresses. This means wasting time calculating the fitness of a large number of particles in the early stages of optimization. To prevent these useless calculations, the proposed algorithm starts working with a small number of particles. An increased number of particles is always a function of the environmental conditions, including an increased number of optima, the modified environmental conditions, or the reduced success index, indicating the optimal progress of the algorithm. In contrast, thanks to stable environmental conditions, the algorithm attempts to reduce the number of particles, which in turn reduces the computational load and increases the optimization speed.

Environmental changes in dynamic environments are not always detectable. For example, only one zone of the entire search space may change, or changes in noisy environments may not be easily detected. In this case, the performance of environmental change detection algorithms is fundamentally impaired. To address this shortcoming, the AidPSO algorithm is designed so that there is no need to detect changes in the environment and always adapt to environmental conditions.

The performance of the proposed algorithm in solving the MPB problem has been investigated as one of the most famous benchmark functions in dynamic environments. Extensive experiments for different MPB problem settings, including a different number of peaks and the length/frequency of the modified environment, indicate the acceptable performance of the algorithm compared to other algorithms in the field of dynamic optimization.

For future research, it is recommended to employ the proposed algorithm to optimize real-world problems. Furthermore, the use of the proposed algorithm in dynamic clustering such as web data is considered. Using self-adaptive mechanisms for the structural parameters of the algorithm can lead to the ability of the method to adapt more quickly to environmental conditions.

References

[1] Mavrovouniotis, Michalis, Changhe Li, and Shengxiang Yang. "A survey of swarm intelligence for dynamic optimization: Algorithms and applications." *Swarm and Evolutionary Computation* 33 (2017): 1-17.

[2] Nguyen, Trung Thanh, Shengxiang Yang, and Juergen Branke. "Evolutionary dynamic optimization: A survey of the state of the art." *Swarm and Evolutionary Computation* 6 (2012): 1-24.

[3] Mavrovouniotis, Michalis, and Shengxiang Yang. "Adapting the pheromone evaporation rate in dynamic routing problems." In *European Conference on the Applications of Evolutionary Computation*, pp. 606-615. Springer, Berlin, Heidelberg, 2013.

[4] Baykasoğlu, Adil, and Fehmi Burcin Ozsoydan. "An improved firefly algorithm for solving dynamic multidimensional knapsack problems." *Expert Systems with Applications* 41, no. 8 (2014): 3712-3725.

[5] Wy, Juyoung, and Byung-In Kim. "A hybrid metaheuristic approach for the rollon–rolloff vehicle routing problem." *Computers & Operations Research* 40, no. 8 (2013): 1947-1952.

[6] Demirtaş, Yonca Erdem, Erhan Özdemir, and Umut Demirtaş. "A particle swarm optimization for the dynamic vehicle routing problem." In *2015 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)*, pp. 1-5. IEEE, 2015.

[7] Khouadjia, Mostepha R., Briseida Sarasola, Enrique Alba, Laetitia Jourdan, and El-Ghazali Talbi. "A comparative study between dynamic adapted PSO and VNS for the vehicle routing problem with dynamic requests." *Applied Soft Computing* 12, no. 4 (2012): 1426-1439.

[8] Okulewicz, Michał, and Jacek Mańdziuk. "Application of particle swarm optimization algorithm to dynamic vehicle routing problem." In *International Conference on Artificial Intelligence and Soft Computing*, pp. 547-558. Springer, Berlin, Heidelberg, 2013.

[9] Mavrovouniotis, Michalis, and Shengxiang Yang. "Applying ant colony optimization to dynamic binary-encoded problems." In *European Conference on the Applications of Evolutionary Computation*, pp. 845-856. Springer, Cham, 2015.

[10] Boryczka, Urszula, and Łukasz Strąk. "Diversification and entropy improvement on the DPSO algorithm for DTSP." In *Asian Conference on Intelligent Information and Database Systems*, pp. 337-347. Springer, Cham, 2015.

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

[12] Mavrovouniotis, Michalis, and Shengxiang Yang. "Memory-based immigrants for ant colony optimization in changing environments." In *European Conference on the Applications of Evolutionary Computation*, pp. 324-333. Springer, Berlin, Heidelberg, 2011.

[13] Mavrovouniotis, Michalis, and Shengxiang Yang. "Interactive and non-interactive hybrid immigrants schemes for ant algorithms in dynamic environments." In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1542-1549. IEEE, 2014.

[14] Gao, Shangce, Yirui Wang, Jiujun Cheng, Yasuhiro Inazumi, and Zheng Tang. "Ant colony optimization with clustering for solving the dynamic location routing problem." *Applied Mathematics and Computation* 285 (2016): 149-173.

[15] Mavrovouniotis, Michalis, and Shengxiang Yang. "Ant colony optimization with memory-based immigrants for the dynamic vehicle routing problem." In *2012 IEEE Congress on Evolutionary Computation*, pp. 1-8. IEEE, 2012.

[16] Mavrovouniotis, Michalis, and Shengxiang Yang. "Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors." *Applied Soft Computing* 13, no. 10 (2013): 4023-4037.

[17] Mavrovouniotis, Michalis, Shengxiang Yang, and Xin Yao. "Multi-colony ant algorithms for the dynamic travelling salesman problem." In *2014 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)*, pp. 9-16. IEEE, 2014.

[18] Van Veen, Barry, Michael Emmerich, Zhiwei Yang, Thomas Bäck, and Joost 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*, pp. 1-10. Springer, Berlin, Heidelberg, 2013.

[19] Yang, Zhiwei, Michael Emmerich, and Thomas Bäck. "Ant based solver for dynamic vehicle routing problem with time windows and multiple priorities." In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2813-2819. IEEE, 2015.

[20] Melo, Leonor, Francisco Pereira, and Ernesto Costa. "Multi-caste ant colony algorithm for the dynamic traveling salesperson problem." In *International Conference on Adaptive and Natural Computing Algorithms*, pp. 179-188. Springer, Berlin, Heidelberg, 2013.

[21] Melo, Leonor, Francisco Pereira, and Ernesto Costa. "Extended experiments with ant colony optimization with heterogeneous ants for large dynamic traveling salesperson problems." In *2014 14th International Conference on Computational Science and Its Applications*, pp. 171-175. IEEE, 2014.

[22] Boryczka, U., & Strąk, Ł. (2015). Heterogeneous DPSO algorithm for DTSP. In *Computational Collective Intelligence* (pp. 119-128). Springer, Cham.

[23] Okulewicz, Michał, and Jacek Mańdziuk. "Two-phase multi-swarm PSO and the dynamic vehicle routing problem." In *2014 IEEE Symposium on Computational Intelligence for Human-like Intelligence (CIHLI)*, pp. 1-8. IEEE, 2014.

[24] Mavrovouniotis, Michalis, and Shengxiang Yang. "A memetic ant colony optimization algorithm for the dynamic travelling salesman problem." *Soft Computing* 15, no. 7 (2011): 1405-1425.

[25] Mavrovouniotis, Michalis, and Shengxiang Yang. "Dynamic vehicle routing: A memetic ant colony optimization approach." In *Automated scheduling and planning*, pp. 283-301. Springer, Berlin, Heidelberg, 2013.

[26] Mavrovouniotis, Michalis, Felipe M. Müller, and Shengxiang Yang. "Ant colony optimization with local search for dynamic traveling salesman problems." *IEEE transactions on cybernetics* 47, no. 7 (2016): 1743-1756.

[27] Li, Changhe, and Shengxiang Yang. "A comparative study on particle swarm optimization in dynamic environments." In *Evolutionary Computation for Dynamic Optimization Problems*, pp. 109-136. Springer, Berlin, Heidelberg, 2013.

[28] Parrott, Daniel, and Xiaodong Li. "Locating and tracking multiple dynamic optima by a particle swarm model using speciation." *IEEE Transactions on Evolutionary Computation* 10, no. 4 (2006): 440-458.

[29] Yang, Shengxiang, and Changhe Li. "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments." *IEEE Transactions on Evolutionary Computation* 14, no. 6 (2010): 959-974.

[30] Li, Changhe, and Shengxiang Yang. "A clustering particle swarm optimizer for dynamic optimization." In *2009 IEEE congress on evolutionary computation*, pp. 439-446. IEEE, 2009.

[31] Branke, Jürgen. "Memory enhanced evolutionary algorithms for changing optimization problems." In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 3, pp. 1875-1882. IEEE, 1999.4.

[32] Jin, Yaochu, and Jürgen Branke. "Evolutionary optimization in uncertain environments-a survey." *IEEE Transactions on evolutionary computation* 9, no. 3 (2005): 303-317.

[33] Bonilla-Vera, José Alberto, Jaime Mora-Vargas, Miguel González-Mendoza, Iván Adrian López-Sánchez, and César Jaime Montiel-Moctezuma. "Brief review of techniques used to develop adaptive evolutionary algorithms." *The Open Cybernetics & Systemics Journal* 11, no. 1 (2017).

[34] Branke, Jürgen, Thomas Kaußler, Christian Smidt, and Hartmut Schmeck. "A multi-population approach to dynamic optimization problems." In *Evolutionary design and manufacture*, pp. 299-307. Springer, London, 2000.

[35] Du, Weilin, and Bin Li. "Multi-strategy ensemble particle swarm optimization for dynamic optimization." *Information sciences* 178, no. 15 (2008): 3096-3109.

[36] Liu, Xiao-Fang, Yu-Ren Zhou, Xue Yu, and Ying Lin. "Dual-archive-based particle swarm optimization for dynamic optimization." *Applied Soft Computing* 85 (2019): 105876.

[37] Blackwell, Tim, and Jürgen Branke. "Multiswarms, exclusion, and anti-convergence in dynamic environments." *IEEE transactions on evolutionary computation* 10, no. 4 (2006): 459-472.

[38] Del Amo, Ignacio G., David A. Pelta, and Juan R. González. "Using heuristic rules to enhance a multiswarm PSO for dynamic environments." In *IEEE congress on evolutionary computation*, pp. 1-8. IEEE, 2010.

[39] Simões, Anabela, and Ernesto Costa. "Evolutionary algorithms for dynamic environments: Prediction using linear regression and Markov chains." In *International Conference on Parallel Problem Solving from Nature*, pp. 306-315. Springer, Berlin, Heidelberg, 2008.

[40] Kennedy, James, and Russell Eberhart. "Particle swarm optimization." In *Proceedings of ICNN'95-International Conference on Neural Networks*, vol. 4, pp. 1942-1948. IEEE, 1995.

[41] Shi, Yuhui, and Russell Eberhart. "A modified particle swarm optimizer." In *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*, pp. 69-73. IEEE, 1998.

[42] Lung, Rodica Ioana, and Dumitru Dumitrescu. "A collaborative model for tracking optima in dynamic environments." In *2007 IEEE Congress on Evolutionary Computation*, pp. 564-567. IEEE, 2007.

[43] Bird, Stefan, and Xiaodong Li. "Using regression to improve local convergence." In *2007 IEEE Congress on Evolutionary Computation*, pp. 592-599. IEEE, 2007.

[44] Blackwell, Tim, Jürgen Branke, and Xiaodong Li. "Particle swarms for dynamic optimization problems." In *Swarm Intelligence*, pp. 193-217. Springer, Berlin, Heidelberg, 2008.

[45] Rossi, Claudio, Mohamed Abderrahim, and Julio César Díaz. "Tracking moving optima using Kalman-based predictions." *Evolutionary computation* 16, no. 1 (2008): 1-30.

[46] Kumar, Priyadarshi Biplab, Dayal R. Parhi, and Chinmaya Sahu. "An approach to optimize the path of humanoid robots using a hybridized regression-adaptive particle swarm optimization-adaptive ant colony optimization method." *Industrial Robot: the international journal of robotics research and application* (2019).

[47] Liu, Xiao-Fang, Yu-Ren Zhou, Xue Yu, and Ying Lin. "Dual-archive-based particle swarm optimization for dynamic optimization." *Applied Soft Computing* 85 (2019): 105876.

[48] Liu, Xiao-Fang, Yu-Ren Zhou, Xue Yu, and Ying Lin. "Dual-archive-based particle swarm optimization for dynamic optimization." *Applied Soft Computing* 85 (2019): 105876.

[49] Jana, Bappaditya, Moumita Chakraborty, and Tamoghna Mandal. "A task scheduling technique based on particle swarm optimization algorithm in cloud environment." In *Soft Computing: Theories and Applications*, pp. 525-536. Springer, Singapore, 2019.

[50] Zhu, Zhen, Long Chen, Chaochun Yuan, and Changgao Xia. "Global replacement-based differential evolution with neighbor-based memory for dynamic optimization." *Applied Intelligence* 48, no. 10 (2018): 3280-3294.

[51] Mukherjee, Rohan, Gyana Ranjan Patra, Rupam Kundu, and Swagatam Das. "Cluster-based differential evolution with crowding archive for niching in dynamic environments." *Information Sciences* 267 (2014): 58-82.