J. Basic. Appl. Sci. Res., 2(4) 4266-4274, 2012
© 2012, TextRoad Publication

ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

A Modified Particle Swarm Optimization Using FCM for Moving Peaks Benchmark

Iman Rezazadeh[2], Sajjad Ghatei[2] and Ahmad Naebi[3]
[1] Department of Computer Engineering, Meshkin Shahr Branch, Islamic Azad University, Meshkin Shahr, Iran
[2,3] Department of Computer Engineering, Ahar Branch, Islamic Azad University, Ahar, Iran

ABSTRACT

Many optimization problems in real world are dynamic and they are changing over time. For resolving these problems, many different algorithms have been proposed. One of these, is PSO algorithm which has well supported its ability in resolving static problems. But this algorithm has some problems in dynamic environments. In this paper, an improved PSO algorithm with inertia parameter has been proposed for dynamic environments which increase the convergence speed of algorithm in getting close toward optimizations. In the proposed algorithm, in order to prevent excessive compression of groups at the end of each iteration, the distance between each group is measured and if this distance is lower than a threshold which is adjusted by a dynamic clustering, the worse group will be eliminated. When some changes is observed in the environment, first the particles' memory is evaluated, then the particles are distributed inside a super globe with the best particle in the center to increase the group diversity. In order to optimize the results, a local search is applied around the best particle of the group. For evaluating the proposed algorithm, moving peaks benchmark was used. The findings showed that the proposed method operates better than other methods.

KEY WORDS: Dynamic Environments, PSO, Moving Peaks Benchmark, FCM

1. INTRODUCTION

PSO is relatively a new heuristic search method. This algorithm has successfully been utilized in variety of applications such as pattern recognition, image processing, machine learning, etc. PSO is an optimized algorithm which is inspired from social and group life of animals like birds in order to get the optimum solution. In PSO a group of particles are located in search area. Giving that each particle presents a candidate solution of the optimization problem. Location of each particle is driven from the best location which has ever met (in terms of self-experience) and location of the best neighborhood particles (in terms of neighborhood experience).
The applications in which the evolutionary algorithms are applied are divided in to two parts: Static and Dynamic. Majority of the real world problems have dynamic nature and are subjected to change over the time. For example, the new tasks that are received continuously and must be scheduled. Parameters which effect the dynamic environment, the frequency of the change, severity of the change, predictability of the change, Cycle length and cycle accuracy. Dynamic environments are divided in to four sections based on defined settings: constant (identical change in each cycle), periodical, homogeneous and alternating [6].
Drawbacks of The PSO in dynamic environments are: old memory and diversity loss which are explained in next paragraphs as bellow:
In the case of any change in environment, the particle's memory is not true anymore and can have a very bad effect on search process. This problem can be solved by two methods: either re-evaluating the memory or forgetting the memory. In re-evaluating, the memory of each particle is verified in each stage. And in forgetting the memory the current location of each particle is replaced with its memory and overall optimization is updated accordingly. Diversity loss occurs when the swarm converges on a few peaks in the landscape and loses its ability to find new peaks, which is required after the environment changes. There are two approaches to deal with diversity losing problem. In the first approach, a diversity maintenance mechanism runs periodically (or when a change is detected) and re-distributes the particles if the diversity falls below a threshold. In the second approach, diversity is always monitored and as soon as it falls below a threshold, the swarm will be re-diversified.
The rest of the paper is organized as follows. In Section 2, related works on dynamic environments are reviewed, in section 3, the proposed algorithm is presented. In Section 4, presents the experimental results of the proposed algorithm along with comparison with alternative approaches from the literature. Finally section 5 includes the conclusion of the present paper.

***Corresponding Author:** Iman Rezazadeh, Department of Computer Engineering, Meshkin Shahr Branch, Islamic Azad University, Meshkin Shahr, Iran. Email: i_rezazadeh@meshkin-iau.ac.ir

2. RELATED WORKS

MPSO is suggested by Blackwell and Branke [4, 5]. The particles in MPSO are divided to M independent groups. Each group contains a fixed number of particles. Information sharing in each group is done in global manner. This mechanism keeps the diversity in two levels: group is divided into sub-groups which penetrate in different sections of search area (diversity between the groups). And each sub-group contains some quantum particles which provide diversity inside the group. In [10], the effectiveness of this algorithm is analyzed and demonstrated that the quantum articles used in this algorithm are only useful when the environment is subject to change and doesn't have much efficiency in other cases.

In [11] because of less efficiency of quantum particles, two types of strategies are utilized: in the first strategy, in the time of detecting any change in environment; the particles are divided to three parts: the first part remains without change, the second part like quantum particles in a cloud are assigned by value of centrality of the best particle and radius of "*r*", and the third part are distributed in whole of the environment. In second strategy useless swarms are detected using fuzzy logic and stopped in order not to waste the system's resources. In [12] a Cauchy Mutation is utilized for detecting the changes in environment, instead of quantum particles. Also the small neighborhoods are used inside the groups. In this work, some of the particles are kept away from center when the group is going to be convergent in order to keep the diversity.

SPSO [5] distributes the particles dynamically between the types. SPSO is extended based on the theory of the types. The limitation of the types depends on parameter r_s which represents the measured radiuses in Euclidean distance from the center of types to its borders. The center of a type so called the "*type's core*", usually is a particle with the best fitness. All of the particles within the r_s radius from the core are categorized as a similar type.

In FMSO [7], two search algorithm, one in parent group and another in child group are utilized. The global search function is utilized in parent group in order to keep the diversity and finding the probable areas in search environment. Mean while hand the child groups are used as local explorer. FMSO starts with a parent group which does the search function in environment in each level if the best founded location in parent's group is improved a child group in centrality of the founded location and radius of r_s from this location is created. Kamosi improved this procedure in [9].

Hashemi and Meybodi introduced cellular PSO, a hybrid model of cellular automata and PSO [3]. In cellular PSO, a cellular automaton partitions the search space into cells. At any time, in some cells of the cellular automaton a group of particles search for a local optimum using their best personal experiences and the best solution found in their neighborhood cells. To prevent losing the diversity, a limit on the number of particles in each cell is imposed.

In KPSO [1], in order to divide the problem into sub-problems consecutive repetition, all particles in problem domain are clustered and each cluster performs the search process independently. At first in this mechanism, the grouping process is performed and stays unchanged for several repetitions. So, there will be enough time for algorithm to do the search. Also because of utilization of the clustering, spatial location of the particles in various groups are considered. Drawback of this approach is defining the suitable number of clusters. In [8], Lee and Yong cluster the particles using fuzzy clustering and grouped the particles using the result of the clustering then according to the progress of the algorithm compound the clusters. If the particles of two clusters are a few, and near to each other, these two clusters are compound. Also in order to solve the problem of two steps forward and two steps back in PSO, when the best particles are updating, the dimensions of that particles are updated one by one. In [13] Lee and Yong, at first all of the particles are distributed in the environment and in each repetition the neighboring particles make one group. This process continues until there is no single particle group in the environment. If two groups are closer than a threshold, then they compound together. And if the number of particles in one group is so many then the worse particles are deleted. If any changes have been made in the environment, again a series of the particles are produced in order to keep the diversity of the particles.

3. PROPOSED METHOD

In our proposed model, the inertia weight has been adjusted adaptively. If the particles of swarm have been improved in previous iteration, it shows that the previous movement of the particles is good and they should continue their pervious movement. So the inertia weight must be high.

In order to improve the performance of PSO algorithm, the search type of each group is specified according to its progress. Thus, first the groups are arranged in descending order. Then each group will have a rating. The first group in the list has a rating of 1 and the last group at the end of the list has rating of n. The group with minimum rating is closer to global optimization, hence it's better to execute a local search to optimize the results and groups with higher rating are far from global optimization so it's better they execute global search. An inertial parameter is used for determining a search to be local or global. The great value for inertial parameter results in global search and small value for inertial parameter leads to local search. For example, in figure 1, there are two different groups, one is blue and the other one is green. As it may be seen, the blue group has a better fitting value so it may be closer to global optimization; hence, it's better to perform a more local search, whereas the green group has lower fitting and it is likely to be far from the global optimization and must perform a more general search, perhaps it may find a better peak and if the optimization found by other groups is not the global optimization, this group may have the ability to explore the global optimization. The rating of each group is used for specifying the minimum amount of inertial weight.

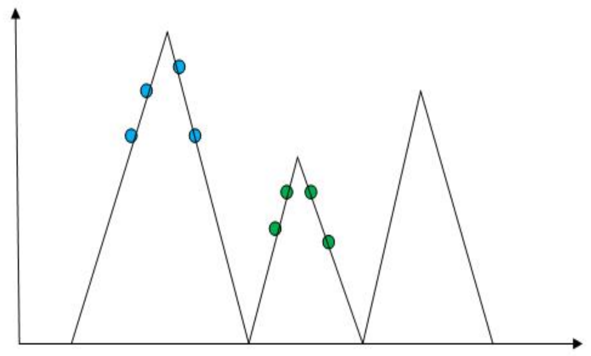


FIGURE 1. Diagram of two groups on peaks

On the other hand, if a group has had a good progress in the previous movement, it indicates that the previous movement of particles of the group was appropriate, so it is better that group particles follow more from their previous speed. For this, the exact amount of inertial is specified according to the group rating and progress.

The groups are divided in to free and converged. If there is no free group in the environment, a group is added to the available groups. Since searching an area is not useful with more than a group, at the end of each iteration, the distance between each two groups are studied to specify whether they are inside each other's radius or not. If two groups are inside each other's radius, it means that the distance between their best found position and the specified threshold, r_{excl} (disposal radius between groups), is less, and the group which *gbest* has the lower fitting is excluded from the search space; this causes that no more than one group to be placed on an optimization. For determining the convergence of a group, the convergence radius, r_{conv} is used which is considered equal to half of the disposal radius value.

For specifying the proper value of disposal radius of particles in each iteration, the available particles are clustered, then the distance between the center of clusters are measured, then the amount of disposal radius is adjusted through 1 and 2 formulas.

$$\min\_dist_i = \min(dist(center_i, center_j)),\ \ where\ \ 1 < i,\ \ j < n,\ \ i \neq j \tag{1}$$

$$r_{excl} = \frac{mean(\min\_dist)}{2^{-n+2}} \tag{2}$$

When a change is observed in the environment, the particles memories are reevaluated, then the position of each particle in group would be equal to a random position inside a super globe with the best found position in the group as the center and the radius of *r*. Then the new position of particles are evaluated and the best individual and group experiments are updated. In this algorithm, in order to improve the results, a local search is performed around the best particle of the best group to improve the best particle of group. The amount of inertial weight, speed and new position will be calculated according to 3 to 6 equations.

$$w^i{}_{\min} = \frac{rank_i}{swarm_{size}} * w_{\max} \tag{3}$$

$$w_i = w_{\min} + (w_{\max} - w_{\min}) * \left(\frac{number\ of\ improved\ \ particle\ in\ swarm_i}{swarm_{size}}\right) \tag{4}$$

$$v_i(t+1) = wv_i(t) + c_1 r_1 (pbest_i - p_i) + c_2 r_2 (gbest_i - p_i) \tag{5}$$

$$p_i(t+1) = p_i(t) + v_i(t+1) \tag{6}$$

If the particles of swarm have been failed, it shows that their previous movement isn't good enough and it is better that these particles don't continue the previous movement so the inertia weight must be decreased. But if all swarms utilize only this method, they may fall in local optimum. To prevent this, we must not let algorithm reduce all swarms inertias weight more than enough.

The groups that have better fitness may be closer to global optimum. So these groups have to have low inertial weight to do local search for groups that have worst fitness may be far from global optimum and therefore may fall in local optimum. In order to avoid from local optimum and find the global optimum, they should have bigger inertial weight to do global search. In this way, the group descending sorted and ordered with number at first to last. First group has the order of one and the last group has the maximum order. The rank of each group is divided to the number of groups and selected as the minimum of group inertial weight. So this way, inertial weight of each group is calculated according to (7), (8):

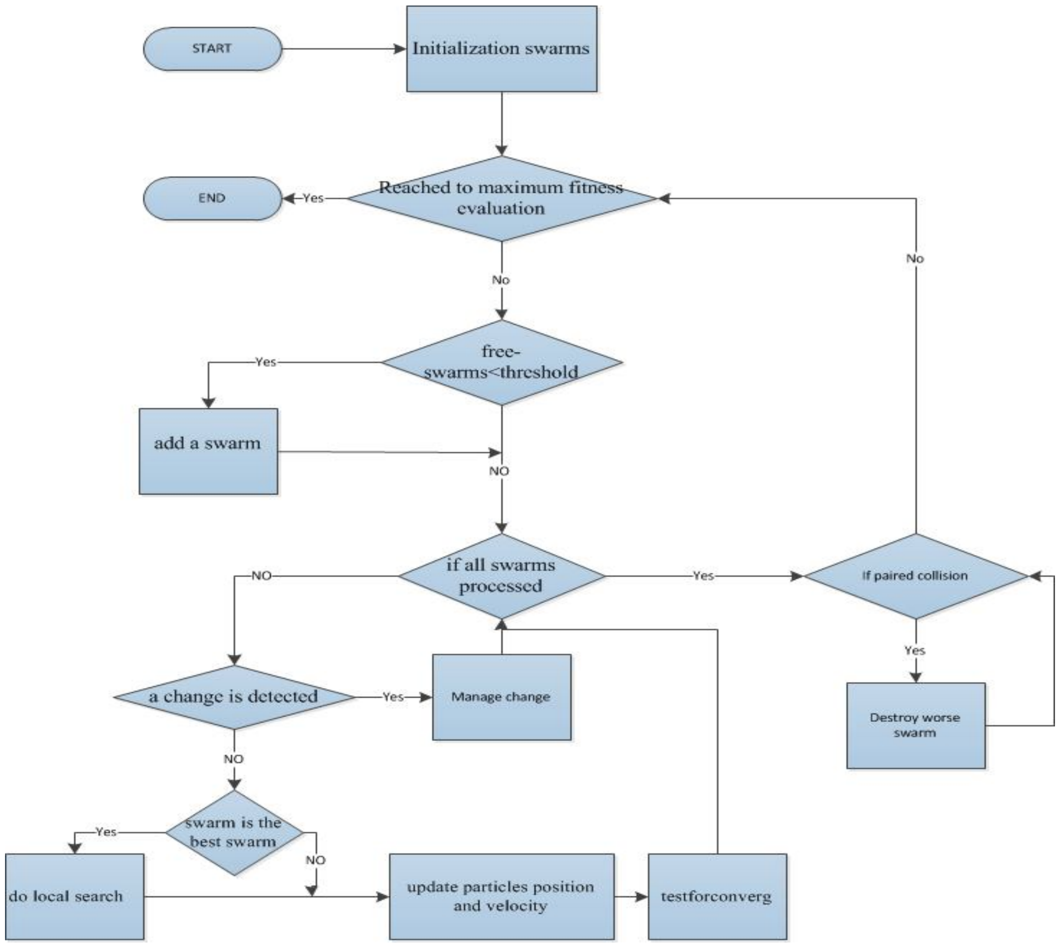


FIGURE 2. Flowchart of proposed algorithm

$$w^i{}_{\min} = \frac{rank_i}{swarm_{size}} * w^i{}_{\max} \tag{7}$$

$$w_i = w^i{}_{\min} + (w_{\max} - w^i{}_{\min}) * (\frac{number\ of\ improved\ particle\ in\ swarm_i}{swarm_{size}}) \tag{8}$$

At first the groups are generated randomly, and start the search. Each group is composed with some PSO particles. Since small neighborhoods causes reduction of convergence's speed, and increase in diversity, performs well in complex environments, this algorithm utilize the small neighborhoods. The less the number of particles for the fixed number of groups, the less the number of evaluations which is leading to keep the environment unchanged for more iterations and effective search will be performed in environment. The groups are categorized to two categories: Converged and Free. If the numbers of free groups in the environment are less than a threshold, one group will be added to the existing groups, and if this number be more than a threshold, the worst group deleted from search domain. At each iteration, velocity and position of a particle *i* in each swarm is updated using its best personal position (*pbest*$_i$) and the best position found by the swarm (*gbest*) according to (5) and (6), respectively. If the fitness of the new position of particle *i* is better than its best personal position (*pbest*$_i$), *pbest*$_i$ will be updated to the new position. Likewise, the best position found in the swarm (*gbest*) will be updated.

Since searching an area with more than one swarm is not very useful, at the end of each iteration every two swarms are checked whether they are searching in the same area or not. Two swarms will be searching in the same area or they are colliding, if the Euclidian distance between their attractors is less than a specified threshold r_{excl}. If a collision between two swarms is detected, the swarm whose attractor is worse than the others

will be destroyed. In the proposed algorithm, when an environment change is detected, particles in the swarm re-evaluate their best personal position (*pbest*) and the particles in the swarms change their behaviors in the following iteration after a change is detected in the environment. They will set their new positions to a random location in a hyper sphere with radius r_q centered at their swarm's attractor. Then they will update their best personal positions (*pbest*) and update the swarm's attractor.
In previous works exclusion distance adjusted without considering environment situation. In purposed algorithm this value is adjusted with considering environment conditions and particles density. Current particles are clustered and then mean of minimum distance between each cluster with other clusters are calculated and used as the r_{excl} according to (1) and (2). For clustering we use FCM and number of clusters equal to the swarm size.

```
Begin
  Initialization swarms
  Repeat
      Adopt swarms number according to free swarms
      For each swarm do
          If a change is detected in the environment then
              Evaluate all pbests then regenerate particles randomly in a hyper sphere with radius r centered at
              gbest and update pbests
          Else if swarm is the best swarm
              do local search around gbest in hyper sphere with radius r and update only gbest
          End if
          For each particle in swarm do
              Update particle position according to (5) and (6)
              Update pbest and gbest
          End for
          Test for converge
      End for
      $r_{excl}$ = fcm_dist
      For each pair of swarms m and n, m<>n do
          Test for conflict
              If there is delete worse swarm
      End for
  Until a maximum number of fitness evaluations is reached
End.
```

FIGURE 3. Pseudocode of the proposed algorithm.

```
Function fcm_dist
    For all particle do
        $data_i$ = position $particl_i$
    End for
    Center = FCM (data, swarmsize)
    For each center
        $dist_i$ = calculate distance with nearest center
    End for
    Calculate $r_{excl}$ using (2)
End.
```

FIGURE 4. Pseudo code of the FCM distance.

4. EXPERIMENTAL STUDYING

In this section, we first describe moving peaks benchmark [2] on which the proposed algorithms is evaluated. Then, experimental settings are described. Finally, experimental results of the proposed algorithm are presented and compared with alternative approaches from the literature.

4.1. MOVING PEAKS BENCHMARK

Moving peaks benchmark (Fig. 3) [2] is widely used in the literature to evaluate the performance of optimization algorithms in dynamic environments [14]. In this benchmark, there are some peaks in a multidimensional space, where the height, width and position of each peak alter when the environment changes. Unless stated otherwise, the parameters of the moving peaks benchmark are set to the values presented in table 1.

Table 1. Default settings of moving peaks benchmark

Parameter	Value
number of peaks m	10
frequency of change f	every 5000 FEs
height severity	7.0
width severity	1.0
peak shape	cone
shift length s	1.0
number of dimensions D	5
cone height range H	[30.0, 70.0]
cone width range W	[1, 12]
cone standard height I	50.0
search space range A	[0, 100]

In order to measure the efficiency of the algorithms, offline error that is the average fitness of the best position found by the swarm at every point in time (Eq. 9), is used [15].

$$offline_{error} = \frac{1}{T}\sum_{t=1}^{T}\left(F(global_{opt}(t)) - F(swarm_{best}(t))\right) \qquad (9)$$

Where T is the maximum iteration, and $swarm_{best}(t)$ is best position solution by the swarm at iteration t.

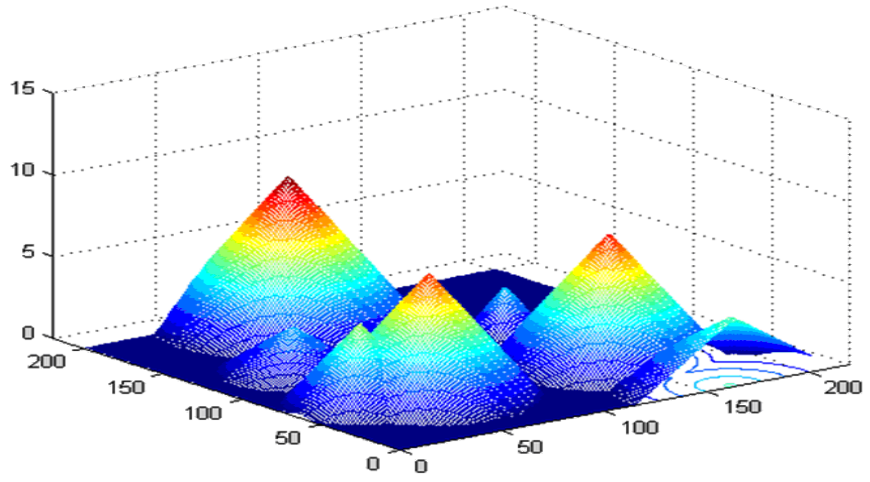
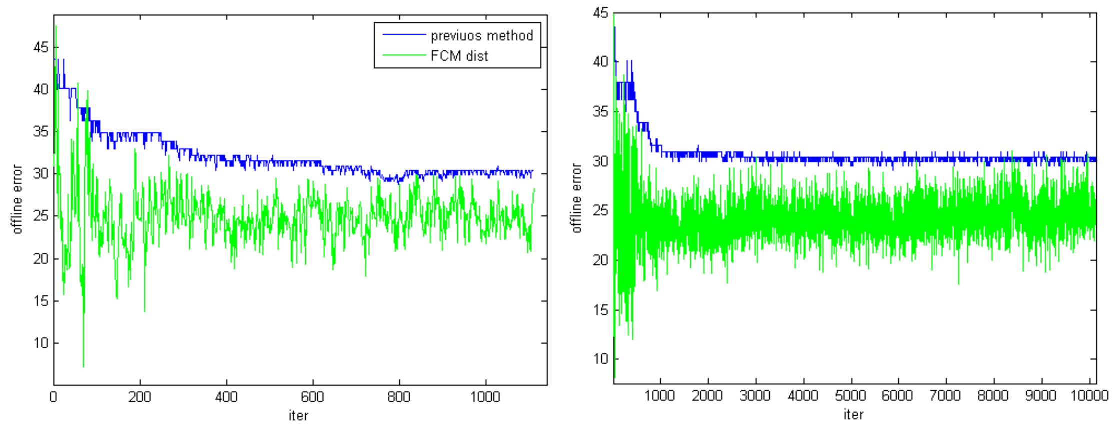


FIGURE 5. Moving peaks benchmark.

4.2. EXPERIMENTAL SETTINGS

For the proposed algorithms the total of acceleration coefficients c_1 and c_2 are set to 1.496180 and the inertial weight w is set to [0, 0.802828]. The number of particles in the swarm is set to 3 particles. The radius of quantum particles (r_q) is set to 0.5. The proposed algorithm is compared with Mqso [5] and FMSO [7] and cellular PSO [3] and kamosi [9]. For mQSO we adapted a configuration 10(5+5q) which creates 10 swarms with 5 neutral (standard) particles and 5 quantum particles with r_{cloud} = 0.5 and r_{excl} = r_{conv} = 31.5, as suggested in [5]. For FMSO, there are at most 10 child swarms each has a radius of 25.0. The size of the parent and the child swarms are set to 100 and 10 particles, respectively [7]. For cellular PSO, a 5-dimensional cellular automaton with 105 cells and moore neighborhood with radius of two cells is embedded into the search space. The maximum velocity of particles is set to the neighborhood radius of the cellular automaton and the radius for the random local search (r) is set to 0.5 for all experiments. The cell capacity θ is set to 10 particles for every cell. Moreover, all particles perform a local search in the iteration after a change in the environment is detected [3]. For the Kamosi algorithms the acceleration coefficients c_1 and c_2 are set to 1.496180 and the inertial weight w is set to 0.729844. The number of particles in the parent swarm and the child swarms (π) are set to 5 and 10 particles, respectively. The radius of the child swarms (r), the minimum allowed distance between two child swarm (r_{excel}) and the radius of quantum particles (r_q) are set to 30.0, 30.0, and 0.5, respectively [9].



(a) **(b)**

Fig. 4. (a) Comparison on FCM_Distance and previous method in environment with 10 peaks and frequency=500. (b) Comparison on FCM_Distance and previous method in environment with 10 peaks and frequency=5000

4.3. EXPERIMENTAL RESULTS

For all algorithms we reported the average offline error and 95% confidence interval for 100 runs. Offline error of the proposed algorithm, mQSO10(5+5q) [5], FMSO [7], cellular PSO [3] and kamosi [9] for different dynamic environment is presented in table 2 to table 6. For each environment, result of the best performing algorithm(s) with 95% confidence is printed in bold. As depicted in the table 2 to table 6, the proposed algorithm outperforms other tested PSO algorithms, including FMSO, for all environments. Moreover, the difference between offline error of the proposed algorithm and the next best algorithm decreases as the environment changes less frequently from *f* = 500 (table 2) to *f* = 10000 (table 6). This is because the proposed algorithm uses less number of particles and so it doesn't waste fitness evaluation and also because the adaptation inertia weight using purposed method, swarm converge very quickly to optimum hence quickly finds better solutions than other algorithms after a change occurs in the environment, especially at the early iterations.

Furthermore, in the proposed algorithm the number of swarms converges to the number of peaks in the environment. This will help the proposed algorithm to track the changes more effectively since there will be a swarm on each peak. For adjusting exclusion distance adaptively, algorithm can adjust exclusion distance corresponding to the environment and if peaks are near to each other doesn't delete swarm that converged to those peaks.

Table 2. Offline error ±Standard error for *f* =500

M	Proposed algorithm	Kamosi	mQSO10(5+5^q)	FMSO	Cellular PSO
1	**4.81±0.14**	5.46±0.30	33.67±3.42	27.58±0.94	13.46±0.7
5	**4.95±0.11**	5.48±0.19	11.91±0.76	19.45±0.45	9.63±0.49
10	**5.16±0.11**	5.95±0.09	9.62±0.34	18.26±0.32	9.42±0.21
20	**5.81±0.08**	6.45±0.16	9.07±0.25	17.34±0.30	8.84±0.28
30	**6.03±0.07**	6.60±0.14	8.80±0.21	16.39±0.48	8.81±0.24
40	**6.10±0.08**	6.85±0.13	8.55±0.21	15.34±0.45	8.94±0.24
50	**5.95±0.06**	7.04±0.10	8.72±0.20	15.54±0.26	8.62±0.23
100	**6.08±0.06**	7.39±0.13	8.54±0.16	12.87±0.60	8.54±0.21
200	**6.20±0.04**	7.52±0.12	8.19±0.17	11.52±0.61	8.28±0.18

Table 3. Offline error ±Standard error for *f* =1000

M	Proposed algorithm	Kamosi	mQSO10(5+5^q)	FMSO	Cellular PSO
1	**2.72±0.04**	2.90±0.18	18.60±1.63	14.42±0.48	6.77±0.38
5	**2.99±0.09**	3.35±0.18	6.56±0.38	10.59±0.24	5.30±0.32
10	**3.87±0.08**	3.94±0.08	5.71±0.22	10.40±0.17	5.15±0.13
20	**4.13±0.06**	4.33±0.12	5.85±0.15	10.33±0.13	5.23±0.18
30	**4.12±0.04**	4.41±0.11	5.81±0.15	10.06±0.14	5.33±0.16
40	**4.15±0.04**	4.52±0.09	5.70±0.14	9.85±0.11	5.61±0.16
50	**4.11±0.03**	4.57±0.08	5.87±0.13	9.54±0.11	5.55±0.14
100	**4.26±0.04**	4.77±0.08	5.83±0.13	8.77±0.09	5.57±0.12
200	**4.21±0.02**	4.76±0.07	5.54±0.11	8.06±0.07	5.50±0.12

Table 4. Offline error ±Standard error for f =2500

M	Proposed algorithm	Kamosi	mQSO10(5+5^q)	FMSO	Cellular PSO
1	**1.06±0.03**	1.10±0.06	7.64±0.64	6.29±0.20	4.15±0.25
5	**1.55±0.05**	1.68±0.16	3.26±0.21	5.03±0.12	2.85±0.24
10	**2.17±0.07**	2.33±0.06	3.12±0.14	5.09±0.09	2.80±0.10
20	**2.51±0.05**	2.79±0.10	3.58±0.13	5.32±0.08	3.41±0.14
30	**2.61±0.02**	2.88±0.09	3.63±0.10	5.22±0.08	3.62±0.12
40	**2.59±0.03**	2.86±0.07	3.55±0.10	5.09±0.06	3.84±0.12
50	**2.66±0.02**	2.97±0.06	3.63±0.10	4.99±0.06	3.86±0.10
100	**2.62±0.02**	3.00±0.05	3.58±0.08	4.60±0.05	4.10±0.11
200	**2.64±0.01**	2.99±0.04	3.30±0.06	4.34±0.04	3.97±0.10

Table 5. Offline error ±Standard error for f =5000

M	Proposed algorithm	Kamosi	mQSO10(5+5^q)	FMSO	Cellular PSO
1	**0.53±0.01**	0.56±0.04	3.82±0.35	3.44±0.11	2.55±0.12
5	**1.05±0.06**	1.06±0.06	1.90±0.08	2.94±0.07	1.68±0.11
10	**1.31±0.03**	1.51±0.04	1.91±0.08	3.11±0.06	1.78±0.05
20	**1.69±0.05**	1.89±0.04	2.56±0.10	3.36±0.06	2.61±0.07
30	**1.78±0.02**	2.03±0.06	2.68±0.10	3.28±0.05	2.93±0.08
40	**1.86±0.02**	2.04±0.06	2.65±0.08	3.26±0.04	3.14±0.08
50	**1.95±0.02**	2.08±0.02	2.63±0.08	3.22±0.05	3.26±0.08
100	**1.95±0.01**	2.14±0.02	2.52±0.06	3.06±0.04	3.41±0.07
200	**1.90±0.01**	2.11±0.03	2.36±0.05	2.84±0.03	3.40±0.06

Table 6. Offline error ±Standard error for f =10000

M	Proposed algorithm	Kamosi	mQSO10(5+5^q)	FMSO	Cellular PSO
1	**0.25±0.006**	0.27±0.02	1.90±0.18	1.90±0.06	1.53±0.12
5	**0.57±0.03**	0.70±0.10	1.03±0.06	1.75±0.06	0.92±0.10
10	**0.82±0.02**	0.97±0.04	1.10±0.07	1.91±0.04	1.19±0.07
20	**1.23±0.02**	1.34±0.08	1.84±0.08	2.16±0.04	2.20±0.10
30	**1.39±0.02**	1.43±0.05	2.00±0.09	2.18±0.04	2.60±0.13
40	**1.37±0.01**	1.47±0.06	1.99±0.07	2.21±0.03	2.73±0.11
50	**1.46±0.01**	1.47±0.04	1.99±0.07	2.60±0.08	2.84±0.12
100	**1.38±0.01**	1.50±0.03	1.85±0.05	2.20±0.03	2.93±0.09
200	**1.36±0.01**	1.48±0.02	1.71±0.04	2.00±0.02	2.88±0.07

5. CONCLUSION

In this paper, we proposed a new multi-swarm PSO algorithm for dynamic environments. The proposed PSO adjust exclusion distance considering environmental conditions. In order to do this all of the present particles are clustered and then distances between centers of clusters are used to adjust exclusion distance. In order to improve efficiency of algorithm we used a local search around best particle of best swarm and also inertia weight adjusted according to the swarm progress so convergence of algorithm is accelerated. And adjusting exclusion radius using clustering particle causes algorithm don't delete swarms that converged to the peaks when peaks are near of each other. To prevent redundant search in the same area if two swarms collide the one with the worse fitness will be removed. In addition, to track the local optima after detecting a change in the environment, particles in each swarm temporarily change their behavior to quantum particles and perform a random search around the swarm's attractor. Results of the experiments show that for all tested environments the proposed algorithm outperforms all tested PSO algorithms, the previously presented multi-swarm algorithm with the similar approach.

Results of the experiments show that for all tested environments the proposed algorithm outperforms all tested PSO algorithms, the previously presented multi-swarm algorithm with the similar approach.

REFERENCES

1. Passaro, A. and A., Starita, 2008. Particle swarm optimization for multimodal functions: a clustering approach. Journal of Artificial Evolution and Applications, 2008: 1-15.
2. Branke, J., 1999. Memory enhanced evolutionary algorithms for changing optimization problems. Congress on Evolutionary Computation, 3: 1875-1882.
3. Hashemi, A.B. and M.R., Meybodi, 2009. Cellular PSO: A PSO for dynamic environments. Advances in Computation and Intelligence, 5821: 422-433.
4. Blackwell, T., 2007. Particle swarm optimization in dynamic environments. Evolutionary Computation in Dynamic and Uncertain Environments, Springer, Berlin, Germany, pp: 29-49.
5. Blackwell, T., J., Branke and X., Li, 2008. Particle swarms for dynamic optimization problems. Swarm Intelligence: Introduction and Applications, Springer, Berlin, Germany, pp. 193-217.

6. Branke, J., 2001. Evolutionary optimization in dynamic environments, Springer, Berlin, Germany, pp: 1-208.
7. Li, C. and S., Yang, 2008. Fast multi swarm optimization for dynamic optimization problems. In Fourth International Conference on Natural Computation, 7: 624–628.
8. Li, C. and S., Yang, 2009. A clustering particle swarm optimizer for dynamic optimization. IEEE Congress on Evolutionary Computation, pp: 439–446.
9. Kamosi, M.A., B., Hashemi and M.R., Meybodi, 2010. A new particle swarm optimization algorithm for dynamic environments. LNCS, 6466: 129-138.
10. Ignacio, G.D.A., D.A., Pelta, J.R., Gonzalez and P., Novoa, 2010. An analysis of particle properties on a multi-swarm PSO for dynamic optimization problems. CAEPIA'09 Proceedings of the Current topics in artificial intelligence and 13th conference on Spanish, LNAI 5988: 32-41.
11. Novoa-Hernandez, P., D.A., Pelta and C.C., Corona, 2010. Improvement strategies for multi-swarm PSO in dynamic environments. Nature Inspired Cooperative Strategies for Optimization, 284: 371-383.
12. Hu, C., X., Wu, Y., Wang and F., Xie, 2009. Multi-swarm particle swarm optimizer with cauchy mutation for dynamic optimization problems. ISICA 2009, LNCS 5821: 443–453.
13. Yang, S. and C., Li, 2010. A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments. IEEE Transaction on Evolutionary Computation, 14(6): 959-974.
14. Moser, I., 2007. All currently known publications on approaches which solve the moving peaks problem. Swinburne University of Technology, Melbourne, Australia.
15. Branke, J., H., Schmeck, 2003. Designing evolutionary algorithms for dynamic optimization problems. Advances in Evolutionary Computing: Theory and Applications. Springer-Verlag New York, pp: 239-262.