Optimization in Dynamic Environments Utilizing a Novel Method Based on Particle Swarm Optimization

D. Yazdani[1] and B. Nasiri[2] and R. Azizi[3] and A. Sepas-Moghaddam[2] and M.R. Meybodi[4]

[1]Young Researchers Club and elites,
Mashhad Branch, Islamic Azad University, Mashhad, Iran
d.yazdani@miau.ac.ir

[2]Department of Computer Engineering and Information Technology
Qazvin Branch, Islamic Azad University, Qazvin, Iran
nasiri.babak@qiau.ac.ir , sepasmoghaddam@qiau.ac.ir

[3]Bojnourd Branch ,
Islamic Azad University , Bojnourd, Iran
reza.azizi@bojnourdiau.ac.ir

[4]Department of Computer Engineering and Information Technology
Amirkabir University of Technology, Tehran, Iran
mmeybodi@aut.ac.ir

ABSTRACT

In numerous real world optimization problems, objective function or constraints of the problem can be changed during time. If these undefined situations are occurred in optimization process, this problem is called dynamic. There are several challenges in dynamic environments optimization, so that algorithms designed for optimization in these environments would utilize several mechanisms in order to conquer the challenges. In this paper, a novel algorithm for optimization in dynamic environments is proposed based on particle swarm optimization in which a novel mechanism have been used for improving the performance. In this mechanism, it is tried to increase the ability of local search around optimum with focusing on best found peak in each environments. The results of the proposed approach are evaluated on moving peak benchmarks and are compared with results of several state of the art algorithms. Experimental results show the superiority of the proposed method.

Keywords: Dynamic optimization problems, particle swarm optimization, moving peaks benchmark, dynamic environments, swarm intelligence, Local Search.

Mathematics Subject Classification Number: 68T01.

1 Introduction

In many real world problems, uncertainty is obvious and clear. Swarm intelligence and evolutionary algorithms are used for solving this kind of problems. Up to now, uncertain problems

are divided into four categories: existence of noise in evaluation function, disturbance in design variables, approximation in fitness function and time-dependent fitness function. In this paper, time-dependent fitness function has been considered, which belongs to the most general types of uncertainties.

So far, different methods have been presented for solving dynamic problems using evolutionary algorithms and swarm intelligence methods (Cheng and Yang, 2010) (Blackwell and Branke, 2006). Four prominent challenges of evolutionary and swarm intelligence algorithms that cause inability in direct use of these methods for optimization in dynamic environments are outdated memory (Carlisle and Dozier, 2000), losing diversity (Cobb, 1990), the existence of multiple potential optimums in the problem space and convergence speed (Blackwell, Branke and Li, 2008).

When an environment changes, current obtained solutions in memory are not valid anymore. Forgetting memory and re-evaluating memory approaches could be contributed for solving outdated memory problem (Carlisle and Dozier, 2000). These solutions are utilized for optimization problem in which the memory is used for storing the acquired information of the problem space as well. In forgetting memory approach, the stored position of each solution is replaced with its current positions in the new environment and in the second approach; the stored positions in memory are re-evaluated. In addition, since most evolutionary algorithms and swarm intelligence methods converge to a point because of their characteristic, so group diversity is lost in the environment and in case of changing the environment, convergence to a new goal would be impossible or time consuming and slow. There are various methods for generating or maintaining group diversity in the environment.

Generally, the approaches for solving diversity loss problem could be divided into two groups. The first group methods solve the problem after its occurrence. Up to now, researchers proposed several methods in this regard. In (Cobb, 1990), an adaptive mutation operator, so called Triggered Hyper-Mutation, which was projected as a coefficient in mutation operator was proposed. In (Nanayakkara, Watanabe and Izumi, 1999), a chaotic mutation in form of adaptive was used for the sack of creating diversity in the environment. In addition, a method with variable local search was presented in (Vavak, Jukes, Fogarty et al., 1998), in which the problem of constant size of mutation step was solved for its adaptation. Replacing the previous solutions after environment changes instead of using random solutions was pursued in (Yu and Suganthan, 2009) for increasing diversity. In (Woldesenbet and Yen, 2009), a variable relocation method was proposed which relocates the solutions regarding changing rate of fitness function during environment change. This method was performed with different radius for each solution. In (Hu and Eberhart, 2002), RPSO method was proposed in consideration of randomizing some parts of the solutions or the whole solutions after detecting environment change. In (Yang and Yao, 2005), an algorithm, called Population-Based Incremental Learning (PBIL) was proposed in which a statistical vector with adaptation ability was used in order to generate solutions. In this method, the vector was used for adapting learning rate after environment

change.

In the second approaches, it is tried to keep diversity of the environment (before and after change). Researchers proposed different methods for keeping diversity. In (Kennedy and Mendes, 2002), a pseudo-grid method with neighborhood structure was presented which was called FGPSO. In (Janson and Middendorf, 2004), HPSO was proposed which was a hierarchical structure for keeping diversity. In addition, a method for optimization in dynamic environments in (Hashemi and Meybodi, 2009a) was proposed, so called Cellular Pso, with respect to utilizing local information exchange property as well as distribution property of cellular automata. In (Hashemi and Meybodi, 2009b), the model proposed in (Hashemi and Meybodi, 2009a) was improved by changing the role of some particles to quantum particles just after changing in the environment. In (Morrison, 2004) , a sentinel placement method was used for keeping diversity. In this method, some sentinels which were distributed in search space were used for creating new population. These sentinels are continuously in the environment and are not removed. Thus, they could be employed for recognizing changes in the system. In (Grefenstette et al., 1992), random immigrant method was proposed in which some random solutions were added to population in order to keep diversity. In (Bui, Abbass and Branke, 2005), a multi objective method was presented in which two objective functions were considered. The first function was the main object and the second one was an objective function to keep diversity in the environment. Finally, in (Andersen, 1991), a method based on fitness sharing was proposed for the sack of keeping diversity.

There are several dynamic problems involve different properties. One of the most important dynamic problems is the one in which there exist several peaks whose height, width and position are changed after each environment change. The simplest way for reacting to these types of environments is considering any changes as an entrance of a new optimization problem which has to be solved again. If there is enough time, this solution is a proper option, but time is often rather short for re-optimization. A common attempt to accelerate optimization process after a change is to use related information of the search space before change to improve the search after change. For instance, if it is supposed that a new optimal point should be near a previous optimal one, it can limit the search to the previous optimal neighbor point. Whether reusing previous information is suitable or not depends on the change characteristic.

If the change is considerable and there is little similarity between pre- and post-change in the environment, restarting optimization algorithm can be the only proper option and any new use of collected information based on the problem space before change will be misleading. In most real world problems, it is hoped that changes are rather gradual, which means there should be a relation between the problem environment before changing and the problem environment after changing so that it enables us to use previous information for improving optimization process. The fundamental question discussed here is what information should be kept and how this information should be used to accelerate search process after changes in the environment.

Another challenge in solving dynamic problems is that there are several potential optimums in these problems. In fact, there are some peaks in dynamic environments whose width, height and location may vary after the environment change in different problems. Therefore, in such problems, each of these peaks can be modified to a global optimum after the environment change. As a result, the algorithms designed for solving these problems should have the ability to cover all peaks so that whenever one of them is transformed to a global optimum, they can find it fast.

Multiswarming is an appropriate way to resolve this problem. In (Blackwell and Branke, 2006) there are some predetermined groups which consist of some agents and every group has to cover one peak. The problem of this method is that if the number of peaks is not equal to the number of groups, algorithm efficiency will decrease. In (Blackwell et al., 2008), Number of groups was determined adaptively according to the number of found peaks in the problem space. Regarding this fact that there is time limitation in dynamic problems, increasing convergence speed of optimization algorithm in dynamic environments is considerably important. The designed algorithms for performing optimization process in dynamic environments are usually the extended versions of the base algorithms which were proposed in static environments. Thus, one of the most important factors in the convergence speed of dynamic algorithms is the used base algorithm.

Up to now, for optimization in dynamic environments, various evolutionary and swarm intelligence algorithms have been used such as particle swarm optimization (PSO) algorithm. PSO algorithm was introduced by Kennedy and Eberhart in 1995 (Kennedy and Eberhart, 1995). Different versions of this algorithm have been proposed since then and appropriate improvements have been done on them. This algorithm has been used for optimization in dynamic environments frequently (Li and Yang, 2012) (Li, Branke and Blackwell, 2006).

In this paper, an optimization algorithm based on PSO is proposed which is designed to be used in dynamic environments for satisfying all requirements of such environments. The proposed algorithm utilizes new approaches to reach better results.

In the proposed algorithm, various mechanisms are used. Some mechanisms that were proposed in previous works are utilized with minor changes. However, the most important contribution of the work is proposing a novel mechanism based on quantum particle which leads to a significant improvement of local search ability around the global optimum and subsequently that of performance.

The proposed algorithm was applied on different configurations of moving peak benchmarks (MPB) (Branke, 1999) which are of the most well-known benchmarks of dynamic environments, and its efficiency is compared with various state of the art algorithms. Experimental results show that the performance of the proposed method outperforms that of other algorithms in this domain. This paper is organized as follows. Particle swarm optimization will be discussed in

section 2. Section 3 describes the proposed algorithm. Experiments and their results will be analyzed in section 4 and the final section concludes the paper.

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{1.1}$$

Algorithm 1: Standard PSO

```
begin
    for each Particle i ε [1..N] do
        initialize x_i , v_i
        P_i = x_i
    end
    G = arg max f(P_i)
    repeat
        for each Particle i ε [1..N] do
            update v_i using equation (2.1)
            Check the velocity boundaries.
            update x_i using equation (2.2)
        end
        if f(x_i) > f(P_i) then
            P_i = x_i
        end
        if f(P_i) > f(G) then
            G = P_i
        end
    until stopping criterion is met
end
```

2 PARTICLE SWARM OPTIMIZATION

PSO is one of the swarm intelligence methods and evolutionary optimization techniques which was proposed by Kennedy and Eberhart in 1995 (Kennedy and Eberhart, 1995). PSO was presented according to animals social interactions such as bird flock and fish swarm. In this method, there is a swarm of particles in which each particle shows a feasible solution for an optimization problem. Every particle tries to move toward the final solution by adjusting its path and moving toward the best individual experience and also the best swarm experience.

Suppose the population size is N. For particle i $(1 \leq i \leq N)$ in D-dimension space, current position is $x_i = (x_{i1}, x_{i2}, , x_{iD})$ and velocity is $v_i = (v_{i1}, v_{i2}, , v_{iD})$. During the optimization process, velocity and position of each particle at each step is updated by (2.1) and (2.2):

$$v_{i,j}(t+1) = \chi[v_{i,j}(t) + c_1 R^1_{i,j}(Pbest_{i,j}(t) - x_{i,j}(t)) + c_2 R^2_{i,j}(Gbest_j(t) - x_{i,j}(t))] \tag{2.1}$$

Where, $x_{i,j}$ is component j of particle i. c_1 and c_2 are acceleration coefficients and χ is constriction factor which has a fixed value less than one. R is a random number with uniform distribution in $[0, 1]$. $Pbest_i$ is the best individual experience of particle i and Gbest is the best experience of swarm. PSO is an iterative algorithm and all particles update their position by equations (2.1) and (2.2) in each performance iteration. In each iteration, after all particle positions are updated, Pbest value of all particles and also Gbest value of swarm are updated with respect to new positions.

Depending on how particles move in PSO, particles may leave the search space, which leads to decreased efficiency and algorithm convergence rate. To solve this problem, some limitations are considered for velocity components values. For this reason, after computing velocity by equation (2.1) in each iteration, all its components values would be reviewed in various dimensions. The value of each velocity vector component can be clamped to the range $[-Vmax, Vmax]$ to reduce the likelihood of particles leaving the search space.

Pseudo code of PSO algorithm is shown in Algorithm 1. Here we suppose the optimization problem is maximizing problem.

3 The Proposed Algorithm

In this section, a new algorithm based on PSO with adaptive quantum based local search, called PSO-AQ is presented for optimization in dynamic environments. In the proposed algorithm, there are a predetermined number of swarms. Particles in each swarm update their position based on their Pbest and Gbest of their own swarm. Each swarm could have two active and inactive modes. An active swarm is the one whose particles are working and are searching in the problem environment. An inactive swarm is the one whose particles are stationary and do not perform any activity during algorithm process. Indeed, fitness evaluation is performed for particles in active swarms, however, no fitness evaluation is performed for inactive swarms particles. In the proposed algorithm, the number of active swarms conforms to the number of found peaks by the algorithm.

At first, only one swarm is active in the problem space. After the first swarm converges to a peak, another swarm will be initialized in the search space and then activated. In order to determine whether a swarm is converged or not, Euclidean distances of all pairs of particles in that swarm should be calculated. If all available particles distances of this swarm from each other are less than a determined value, rconv, the swarm has converged. During the execution of the algorithm whenever all active swarms converge, another inactive swarm will be activated after being initialized. If a recently activated swarm converges to a peak which is covered by another active swarm, a race condition occurs. In this condition, the swarm with a better Gbest fitness value continues its activity and the beaten swarm becomes inactive. Two swarms converge to a peak when Euclidean distance of their Gbest positions is less than a specified

value rexcl. Therefore, when all active swarms have converged, an inactive swarm is initialized in the problem space and becomes active and when two swarms converge to a peak one of them becomes inactive.

As a result, any swarm that becomes active has the chance to converge to a peak where no swarm has resided yet. But if it converges to a covered peak, it will be inactivated with a high probability, because after approaching the peak, the probability of being placed in a better position than the previously residing swarm is very low. According to this trend, the number of active swarms is always one more than the number of found peaks by the algorithm since other than the residing swarms on peaks, there is always a new activated swarm trying to find an undiscovered peak. Indeed, there is always an active swarm which search for finding possible undiscovered peaks.

To discover change in the environment, at the beginning of algorithm execution, a random point called test point is selected in the problem space and its fitness value will be stored. After executing each iteration of algorithm for all active swarms, fitness value of test point is reevaluated. If the obtained fitness value is not equal to the stored value based on previous fitness evaluation of test point, then the environment has changed. It should be noted that test point position would remain fixed until the end of algorithm. Test point approach has been proposed for such environments in which environments are changed globally. Otherwise, it could not be applicable.

After the environment changes, invalid memory problem has to be resolved first. In the proposed algorithm, after discovering change in the environment, fitness values of all active swarms' particles' Pbest positions are reevaluated and in each swarm Gbest becomes equal to the best Pbest of its particles. So, invalid memory problem is resolved. After discovering changes in the environment, diversity problem in swarms would be solved as well. In fact, this problem occurs, due to this fact that in converged swarms, the density of swarm has increased considerably and particles have been set too much close to each other. On the other hand, Pbest position of particles and Gbest of swarm are also very close to each other and the values of vectors components of particles velocity is very close to zero in this condition. In this case, diversity in swarm is too low and swarm ability is considerably weak to continue optimization process after the environment change and displacement of optimal point. In the proposed algorithm, for increasing diversity in swarms, random values are given to velocity vector components. The range in which particle velocity components are randomly initialized must be based on the length of peaks movements ($vlength$). The length of peaks movements has been omitted from the prior knowledge of the problem in many references. On the other hand, in case of lacking this knowledge, the length of peaks movements could be extracted, based on position of the peaks in consecutive environments.

For example, after the environment change, if $vlength$ is equal to x, the goal is located in a neighborhood of radius x of its previous position, where currently the swarm is concentrating.

Therefore, particles have to search the goal at this distance. Indeed, after the environment change, the goal is located in a d-dimensional ball with the centrality of the goal position before the environment change and with radius x in every dimension. As a result, particles have to be distributed in this d-dimensional ball. Now, if velocity components values of particles are adjusted randomly in $[-x, x]$, after one iteration of algorithm execution, particles are distributed at the mentioned distance. Thus, increase in diversity is done based on peaks shift length. It should be noted that the discussed process is only performed on swarms which have converged. If a swarm is active but has not converged yet, it means that the swarm maintains its diversity and after changes in the environment, no change is needed to increase diversity.

In the proposed algorithm, in order to increase accuracy of obtained results, a novel approach is applied based on quantum particles. In this approach, a quantum particle is used only for the best swarm in order to do a better local search around the possible optimum peak. Quantum particles approach is used for maintaining diversity and enhancing optimization (Blackwell et al., 2008) (Blackwell and Branke, 2006). The idea is that all swarms have a quantum cloud around their Gbest and in each iteration, quantum particles of every swarm are located randomly in this cloud (radius of this quantum cloud remains fixed during the execution of algorithm). Then, if the position of the best quantum particle is better than swarm Gbest, Gbest will become the position of the best quantum particle. In the proposed algorithm, a quantum particle is used in sequential form just for the best swarm. The best swarm is the one which has the best Gbest amongst active swarms according to the Gbest fitness value. With respect to (Blackwell et al., 2008) and (Blackwell and Branke, 2006), value of quantum cloud has to be determined based on peaks shift length, which could be calculated after a number of environment changes.

In the proposed algorithm, in each iteration, there is a quantum particle that can try try_number times to improve Gbest position of the best swarm. In this process, first a random position with a uniform distribution is determined in the quantum cloud which is the quantum particle position. Quantum particle position is obtained by equation (3.1):

$$x_{Quantum,j} = Gbest_{bestSwarm,j} + (r_{cloud} \times R_j) \tag{3.1}$$

In this equation, Rj is a random number with a uniform distribution in $[-1, 1]$. So component jth of quantum particle locates at [-r_{cloud},r_{cloud}] around the same component in Gbest of the best swarm. Therefore, the quantum particle position is placed in a d-dimensional ball around Gbest of the best swarm. Then, fitness value of the quantum particle is compared with the fitness value of Gbest of the best swarm. If Gbest fitness value of the best swarm is better, no change is done in Gbest position of the best swarm. But if quantum particle fitness value is better, then Gbest position of the best swarm is modified based on equation (3.2):

$$Gbest_{bestSwarm} = x_{Quantum} \tag{3.2}$$

This action is performed try_number times based on the Gbest position of the best swarm and each time, if the quantum particle position is better, Gbest position of the best swarm will change by equation (3.2). Therefore, after changing Gbest position based on equation (3.2), new position of the quantum particle is determined by equation (3.1) according to the new Gbest position. So in each iteration and in the best case, Gbest position of the best particle would be improved try_number times by equation (3.2) and in the worst case no changes would occur.

Quantum approach is proposed in order to increase accuracy and convergent rate of the best swarm toward the goal with respect to fitness evaluation However, as it can be observed in equation (3.1), efficiency of this method depends completely on r_{cloud} value. r_{cloud} value must be determined according to $vlength$, but a fixed value of this parameter may decrease quantum particle efficiency; Because, at first, the value of this parameter is appropriately adjusted to the space that has to be searched, but this space decreases gradually while moving toward the goal and after a while r_{cloud} value is much bigger than the space that has to be searched. Consequently, the probability of this quantum particle position obtained by equation (3.1) being better that Gbest position of the best swarm decreases too much. For removing the discussed weakness and to adapt r_{cloud} to the search area, r_{cloud} value should decrease simultaneously with the swarm movement toward the goal. For this reason, r_{cloud} value has to be multiplied by a number less than one in each iteration. We can use various mechanisms to determine this number.

In this paper, a self$-$adaptive mechanism is proposed in consideration of determining this value. In this mechanism, after performing try_number times of equation (3.1), the number of performing equation (3.2), which means the number of improving $Gbest_{best\_swarm}$ position using quantum particle are tracked. The value of r_{cloud} in each iteration is updated by equation (3.3)

$$r_{cloud}(t) = r_{cloud}(t-1) \times (L_{Low} + ((\frac{S}{try\_number}) \times (1 - L_{Low}))) \tag{3.3}$$

In this equation, S indicates the number situations, in which $Gbest_{best\_swarm}$ value is improved in the current iteration using the quantum particle.

S/try_number always is a number in range of [0,1], due to this fact that the amount of S is equal to try-number in the best case. When the value of S/try_number approaches to 1, means that the success rate of quantum particle is high. In this situation, there is no necessity for more decreasing r_{cloud}., since, quantum particle using this cloud involves acceptable efficiency. On the other hand, when the value of S/try_number approaches to zero, it could be concluded that the space of quantum particles search cloud is greater than the space which must be searched by quantum particle for finding better positions. In this case, the probability of occurrence of the situation obtained using equation (3.1) is decreased. Thus, the value of r_{cloud} must be decreased with a greater rate. Equation (3.3), decreases the value of the r_{cloud} in each iteration regarding the defined range.

Algorithm 2: Proposed Algorithm PSO-AQ

```
1  begin
2      for first swarm do
3          activate first swarm;
4          foreach Particle j do
5              initialize x_{1,j} , v_{1,j}
6              Pbest_{1,j} = x_{1,j}
7          end
8          Gbest = arg max f(Pbest_{1,j})
9      end
10     initialize testpoint
11     repeat
12         TestForChange()
13         Exclusion()
14         IsConverged()
15         if all activated swarm are converged then
16             Activation()
17         end
18         foreach activated swarm i do
19             foreach Particle j do
20                 Update X_{i,j}(t+1) and Pbest_{i,j}(t+1)
21             end
22             Gbest = arg max f(Pbest_{i,j})
23         end
24         for cnt=1 to TryNumber do
25             Update Quantum based on equation 3
26             if f(Quantum) > f(Gbest_{BestSwarm}) then
27                 Gbest_{BestSwarm} = Quantum
28             end
29         end
30         Update r_{cloud} based on equation 5
31     until stopping criterion is met
32 end
```

Algorithm 3: TestForChange

```
begin
    Evaluate testpoint
    if new value is different from last iteration then
        Reinitialize r_cloud
        foreach activated swarm i do
            foreach Particle j do
                Reevaluate Pbest_{i,j}
            end
            Gbest = arg max f(Pbest_{i,j})
            if swarm i is converged then
                Set each particle Velocity vector based on vlenght randomly
            end
        end
    end
end
```

Algorithm 4: Exclusion

```
begin
    foreach couple activatedSwarm_{i,j} do
        if Euclidian dist. between Gbest_i and Gbest_j < r_excl then
            if f(Gbest_i) < f(Gbest_j) then
                inactivate swarm i
            else
                inactivate swarm j
            end
        end
    end
end
```

Algorithm 5: IsConverged

```
begin
    foreach activatedSwarm_i do
        if Euclidian dist. between all couple particle m , n is < r_conv then
            swarm i is converged
        end
    end
end
```

Algorithm 6: Activation

```
begin
    for an inactivated Swarm i do
        Activate swarm i
        Initialize x_{i,j} , v_{i,j}
    end
    Pbest_{i,j} = x_{i,j}
    Gbest = arg min f(Pbest_{i,j})
end
```

Thus, r_{cloud} value in each iteration is adaptively in [$r_{cloud}(t-1) \times L_{LOW}$, $r_{cloud}(t-1)$]. It should be mentioned that r_{cloud} value is initialized after every environment change to adapt itself with the problem space conditions. The pseudo code for the proposed algorithm is presented in algorithm 2 and applied functions are shown in algorithm 3 to 6.

4 Experimental Study

In this section, we first describe Moving Peaks Benchmark (MPB)(Branke, 1999) which is the most well-known benchmark for evaluating the performance of optimization algorithms in dynamic environments. Then, we assess accuracy and efficiency of the proposed algorithmand some other state of the art algorithms which are applied on MPB. The main metric for evaluating the performance of algorithms in this domain is offline error which is indicates the average of the best found positions fitness using algorithms during the running optimization process (Branke, 1999). In other word, the value of offline error is the average of current errors. Current error at time t is the deviance of the best found position using algorithm at time t in the current environment and the position of global optimum in the current environment. The value of offline error is equal or greater than zero, where zero indicates the ideal situation. In this paper, we used the source codes presented in (Branke, 1999) in the experiments.

4.1 EXPERIMENTAL SETUP

In this section, we have tested proposed algorithm on Moving Peaks Benchmark (MPB) (Branke, 1999), which is the best-known benchmark in dynamic environments optimization (Jin and Branke, 2005). The reason for its popularity among the researchers is that this benchmark can produce numerous and various conditions and situations of dynamic environments. By utilizing this benchmark, we can study algorithms performances from different aspects.

In order to measure the efficiency of the algorithms, offline error that is the average of the difference between fitness of the best solution found by the algorithm and fitness of the global

optimum is used (Branke, 1999).

$$offline\_error = \frac{1}{FEs}\sum_{t=1}^{FEs}(fitness(gbest(t)) - fitness(globalOptimum(t))) \quad (4.1)$$

Where FEs is the maximum function evaluations, and gbest(t) and globalOptimum(t) are the best position found by the algorithm and the global optimum at the tth fitness evaluation, respectively. In other word, the amount of offline error equals to the average of all current errors which is defined in time t as deviance between the best found position by the algorithm in time t in the current environment and the position of the global optimum in the current environments. The value of Offline error is constantly a non-negative number, where it is zero in the ideal situation.

Experiments were done with respect to MPB parameters which are given in table (1).

Table 1: MPB Parameters Configuration

Parameter	**Value**
Number of peaks, M	Variable between 1 to 200
Change frequency	500,1000,2500,5000,10000
Height change	7.0
Width change	1.0
Peaks shape	Cone
Basic function	No
Shift length, s	1.0
Number of dimensions,D	5
Correlation Coefficinet,	0
peaks location range	[0 100]
Peak height	[30.0 70.0]
Peak width	[1 12]
Initial value of peaks	50.0

Adjustments of parameters values in the proposed algorithm are as follows:
There are 5 particles in each swarm. It was proved in (Blackwell et al., 2008) that this number of particles has the best efficiency in solving MPB problems. At the start of algorithm execution, there is only one active swarm. According to (Blackwell et al., 2008) c1 and c2 are considered 2.05 and value is considered 0.729843788. rexcl and rconv are determined based on (Blackwell and Branke, 2006) (Blackwell et al., 2008). In accordance with many other references in this domain, it is considered that there is a prior knowledge about $vlength$. It was shown in (Blackwell et al., 2008) that adjusting r_{cloud} value equal to the half of $vlength$ (shift severity) leads to appropriate results, so we consider this as the initialized value of r_{cloud} as well.

After finding the environment change, to increase diversity in active swarms, different components values of particles velocity vector are determined randomly in $[-vlength, vlength]$ to expand in the range of the new position of the goal. Experiments were repeated independently 50 times and each time they were performed by different random seeds. Every experiment continued until environment changes reached 100 times. For example when the change frequency was 5000, experiments performed 5105 fitness evaluations and during this time the environment changed 100 times. Table 2 shows the effect of various values of try_number parameter on algorithm efficiency for optimizing MPB with 10 peaks, the change frequency 5000 and severity 1. Experimental results in table 2 were obtained such that r_{cloud} remained constant during the algorithm execution.

Table 2: effect of various values of try_number parameter on algorithm efficiency

try_number	**Offline_error (Standard_error)**
5	1.30(0.06)
10	1.17(0.05)
15	1.13(0.06)
20	1.09(0.07)
25	1.20(0.09)
30	1.32(0.10)
50	1.97(0.14)

As it is observed in the course of experiments, by increasing try_number up to 20, algorithm efficacy improves because of the increase in local search ability around the global optimum position. However,it decreases after a period of increase, and thus the efficiency decreases.

In fact, the degree of executing fitness evaluation for quantum particles grows by try_number increase and it leads to the consumption of high amount of fitness evaluation by that particle and swarms will lose the opportunity for a better search. To be more precise, an environment is supposed in which the change frequency is 5000, fitness evaluations is performed for 10 peaks, with 10 swarms being located in them. Since each of them has five particles, they execute 50 fitness evaluations in total, and each one of the 100 iterations are performed until the environment changes and they move toward the goal. Now if try_number is 50, this chance reduces in half and only 50 iterations can be executed until the next environment change, which means that 2500 fitness evaluations are done in 50 iterations. Therefore, swarms cannot shorten their distance enough from the new position of the peak, which results in swarms efficacy decrease in following the peaks. With a proper value for try_number, and through local search ability increase, algorithm efficiency improves. As it is observed in table 2, when try_number is 20, a better result is obtained. So from now on, try_number is considered 20.

Table 3 shows algorithm efficiency when r_{cloud} is decreased by using equation 5. It shows the effect of different values of L_{LOW} on algorithm performance. Obviously, when L_{LOW} is more

Table 3: The effects of decreasing r_{cloud} on the efficiency of the proposed algorithm

L_{min}	**Offline_error (Standard_error)**
1	1.09(0.07)
0.95	1.06(0.03)
0.9	1.04(0.05)
0.8	0.99(0.08)
0.75	**0.89(0.03)**
0.7	0.92(0.06)
0.65	1.18(0.11)
0.6	1.33(0.10)
0.5	1.70(0.12)

than or equal to 0.7, quantum particle performance is better than the time r_{cloud} is constant (r_{cloud}=1). When L_{LOW} is less than 0.7, quantum particle efficiency, and consequently the algorithm efficiency, decreases. Indeed, in this case, quantum cloud decreases by equation 5 with a higher rate than quantum particle development. Therefore, in this condition, the number of fitness evaluations done by quantum particle not only does not increase efficiency, but also it takes the chance of more searches from particles. According to table 7, the most appropriate CFmin is 0.75.

The results of the table 3 show that how much an appropriate local search around the best peak could improve the algorithms efficiency.

4.2 EXPERIMENTAL RESULTS

In this subsection, the results of the PSO-AQ are compared with other state of the art algorithms in this domain. Some presented results of the comparative algorithms are obtained by implementing and others are extracted from their references. Table 4 indicates the results of five algorithms including mQSO(5,5q) (Blackwell and Branke, 2006), AmQSO (Blackwell et al., 2008), mPSO (Kamosi, Hashemi and Meybodi, 2010), APSO (Rezazadeh, Meybodi and Naebi, 2011) and the proposed algorithm on MPB with the number of peaks of 1,5,10,20,30,50,100,200, environment change frequencies of 500, 1000, 2500 and 10000 and shift severity of 1.

As it can be observed in table 4, the proposed algorithm outperforms other four algorithms in all cases. When the change frequency of the environment is adjusted low, convergence rate of algorithms after the environment change would be important because the environment changes after a short time and in the case of low convergence rate of the algorithm, swarms may miss optimal points and even some of them may never converge to a peak.

Table 4: Comparison of Offline error (Standard error) of the Proposed Algorithm with four other algorithms in Different Change Frequencies (C-F) and Number of Peaks

ALG.	C-F	NUMBER OF PEAKS							
		1	5	10	20	30	50	100	200
mQSO(5,5q)	500	8.19(0.17)	8.54(0.16)	8.72(0.20)	8.80(0.21)	9.07(0.25)	9.62(0.34)	11.91(0.76)	33.67(3.42)
AmQSO		7.48(0.19)	7.34(0.31)	7.57(0.32)	7.10(0.39)	6.82(0.34)	5.37(0.42)	5.77(0.56)	3.02(0.32)
mPSO		8.88(0.14)	8.91(0.17)	8.76(0.18)	8.43(0.17)	8.01(0.19)	7.19(0.23)	6.69(0.26)	8.71(0.48)
APSO		6.20(0.04)	6.08(0.06)	5.95(0.06)	6.03(0.07)	5.81(0.08)	5.16(0.11)	4.95(0.11)	4.81(0.14)
PSO-AQ		**6.13(0.17)**	**5.68(0.15)**	**6.21(0.16)**	**5.43(0.18)**	**5.71(0.18)**	**4.96(0.17)**	**4.27(0.19)**	**2.88(0.19)**
mQSO(5,5q)	1000	5.54(0.11)	5.83(0.13)	5.87(0.13)	5.81(0.15)	5.85(0.15)	5.71(0.22)	6.56(0.38)	18.60(1.63)
AmQSO		5.75(0.26)	4.77(0.45)	6.06(0.14)	5.20(0.38)	5.36(0.47)	4.56(0.40)	2.90(0.32)	2.33(0.31)
mPSO		5.78(0.09)	5.60(0.09)	5.33(0.10)	5.15(0.12)	4.97(0.13)	4.57(0.18)	3.93(0.16)	4.44(0.24)
APSO		4.21(0.02)	4.26(0.04)	4.11(0.03)	4.12(0.04)	4.13(0.06)	3.87(0.08)	2.99(0.09)	2.72(0.04)
PSO-AQ		**3.98(0.09)**	**3.97(0.10)**	**3.87(0.11)**	**3.98(0.13)**	**3.66(0.14)**	**3.13(0.12)**	**2.86(0.12)**	**1.97(0.12)**
mQSO(5,5q)	2500	3.30(0.06)	3.58(0.08)	3.63(0.10)	3.63(0.10)	3.58(0.13)	3.12(0.14)	3.26(0.21)	7.64(0.64)
AmQSO		3.07(0.12)	3.53(0.14)	3.68(0.15)	3.24(0.18)	2.73(0.11)	2.49(0.10)	2.16(0.19)	0.87(0.11)
mPSO		3.36(0.05)	3.31(0.05)	3.26(0.07)	3.15(0.08)	3.07(0.11)	2.66(0.16)	2.04(0.12)	1.79(0.10)
APSO		2.64(0.01)	2.62(0.02)	2.66(0.02)	2.61(0.02)	2.51(0.05)	2.17(0.07)	1.55(0.05)	1.06(0.03)
PSO-AQ		**2.55(0.05)**	**2.65(0.07)**	**2.63(0.07)**	**2.32(0.05)**	**2.14(0.08)**	**1.91(0.13)**	**1.44(0.09)**	**0.73(0.12)**
mQSO(5,5q)	10000	1.71(0.04)	1.85(0.05)	1.99(0.07)	2.00(0.09)	1.84(0.08)	1.10(0.07)	1.03(0.06)	1.90(0.18)
AmQSO		2.52(0.10)	1.89(0.14)	1.55(0.08)	1.78(0.09)	1.28(0.12)	0.76(0.06)	0.45(0.04)	0.19(0.02)
mPSO		1.48(0.02)	1.50(0.03)	1.47(0.04)	1.43(0.05)	1.34(0.08)	0.97(0.04)	0.70(0.10)	0.27(0.02)
APSO		1.36(0.01)	1.38(0.01)	1.46(0.01)	1.39(0.02)	1.23(0.02)	0.82(0.02)	0.57(0.03)	0.25(0.01)
PSO-AQ		**1.29(0.18)**	**1.32(0.06)**	**1.18(0.05)**	**0.98(0.06)**	**0.70(0.05)**	**0.56(0.07)**	**0.35(0.08)**	**0.14(0.00)**

By increasing change frequency, algorithms efficiency is improved because swarms have more chance to reach goals before the environment change. In the proposed algorithm, swarms perform less fitness evaluations than other algorithms at every iteration. In fact, in the proposed algorithm, all swarms, except the best swarm, perform only 5 fitness evaluations at each iteration (equal to the population size of the swarm). It enables swarms to execute more iterations compared with other algorithms before the environment change, particularly when the number of swarms is high. Also, in the proposed algorithm, to find the environment change, just one fitness evaluation is performed for $test\_point$ at each iteration of algorithm execution. In other algorithms, this number is much more than one and is equal to the number of available swarms in the problem at every iteration.

In the proposed algorithm, after covering all peaks, the algorithm performs fitness evaluations in proportion with the number of peaks in the problem space. In addition, in order to increase diversity, a mechanism was proposed without performing any additional fitness evaluations which generates diversity in swarms, well after discovering the environment change. The result in the proposed algorithm is that swarms perform less fitness evaluations and have more iterations to reach goals before the environment change. Also, using quantum idea for the best swarm leads to a better search around the best found position by the algorithm. Indeed, by decreasing quantum cloud radius during the execution of the algorithm, local search ability increases around the goal. In addition, quantum idea in the proposed algorithm is designed in a manner that there would be a probability of Gbest improvement in every fitness evaluation (try_number).

It is worth mentioning that in (Blackwell and Branke, 2006) and (Blackwell et al., 2008), the quantum particles approach was used in order to increase diversity. However, in the proposed algorithm, this approach is used in consideration of a better local search around the optimum

peak which its efficiency is significantly improved thanks to a self-adaptive mechanism.

Figure 1 shows average $current\_error$ and $offline\_error$ graphs of 50 times execution of the proposed algorithm (PSO-AQ) and AmQSO on MPB with 10 peaks, change frequency 5000 and $vlenght$ 1 in scenario 2, for 100 environment changes. As it can be observed, after each environment change, swarms in the proposed algorithm converge fast to the new positions of goals and because of increasing local search ability around the best found position, $current\_error$ value decreases well before the next environment change. A considerable matter in this graph is a fast decrease in $offline\_error$ value after starting execution of the proposed algorithm. In the proposed algorithm, activating and inactivating mechanisms of swarms are designed so that those useless and inefficient swarms would not exist in the problem space. Also, the increase in local search ability is only done in the best swarm by means of quantum particle, and it is not used in other swarms in order not to waste the fitness evaluation. Because, increasing local search ability cannot be useful in non-optimal peaks. Therefore, the proposed algorithm would have a faster convergent rate by better use of fitness evaluation.

Table 5: A Comparison of Offline errors (Standard errors) belong to seven Algorithms on the MPB Problem with Different Shift Lengths

	Shift Severity			
Algorithm	**1.0**	**2.0**	**3.0**	**5.0**
mQSO	1.75(0.06)	2.40(0.06)	3.00(0.06)	4.24(0.10)
AmQSO	1.51(0.10)	2.09(0.08)	2.72(0.09)	3.71(0.11)
mCPSO	2.05(0.07)	2.80(0.07)	3.57(0.08)	4.89(0.11)
SPSO	2.51(0.09)	3.78(0.09)	4.96(0.12)	6.76(0.15)
rSPSO	1.50(0.08)	1.87(0.05)	2.40(0.04)	3.25(0.09)
PSO-CP	1.31(0.06)	1.98(0.06)	2.21(0.06)	3.20(0.13)
PSO-AQ	**0.89(0.03)**	**1.14(0.09)**	**1.86(0.08)**	**2.28(0.12)**

In the other parts of experiments, the performance of proposed algorithm is compared with 6 other algorithms on the MPB problems with 10 peaks, change frequency 5000 and different settings of the shift length ($vlenght$). The experimental results regarding the $offline\_error$ and $standard\_error$ are shown in Table 5.

From Table 5, it can be observed that the results obtained by proposed algorithm PSO-AQ are considerably better than the results in the other six algorithms on MPB problems with different shift severities. It is obvious that the peaks are more and more difficult to be tracked with the increase of the shift lengths. Thus, the performance of all algorithms degrades when the shift length increases. As it is shown in table 5, with the increase in shift severity, the proposed algorithm has a better performance compared to other six algorithms and is robust to the shift severity. The reasons for the appropriate efficiency in the proposed algorithm in

various shift lengths are expanding particles based on shift severities after the environment change and increasing search ability around the best found peak.

Table 6: A Comparison of Offline errors (Standard errors) belong to 14 Algorithms on the MPB Problem with Different Number of Peaks

	Number Of Peaks							
Algorithm	**1**	**5**	**10**	**20**	**30**	**50**	**100**	**200**
mQSO	4.92(0.21)	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)
AmQSO	0.51(0.04)	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)
CLPSO	2.55(0.12)	1.68(0.11)	1.78(0.05)	2.61(0.07)	2.93(0.08)	3.26(0.08)	3.41(0.07)	3.40(0.06)
FMSO	3.44(0.11)	2.94(0.07)	3.11(0.06)	3.36(0.06)	3.28(0.05)	3.22(0.05)	3.06(0.04)	2.84(0.03)
RPSO	0.56(0.04)	12.22(0.76)	12.98(0.48)	12.79(0.54)	12.35(0.62)	11.34(0.29)	9.73(0.28)	8.90(0.19)
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)
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)
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)
mPSO	0.90(0.05)	1.21(0.12)	1.61(0.12)	2.05(0.08)	2.18(0.06)	2.34(0.06)	2.32(0.04)	2.34(0.03)
PSO-CP	3.41(0.06)	-	1.31(0.06)	-	2.02(0.07)	-	2.14(0.08)	2.04(0.07)
RVDEA	1.02(-)	-	3.54(-)	3.87(-)	3.92(-)	3.78(-)	3.37(-)	3.54(-)
SFA	0.42(0.07)	0.89(0.09)	1.05(0.04)	1.48(0.05)	1.56(0.06)	1.87(0.05)	2.01(0.04)	1.99(0.06)
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)**
CLDE	1.53(0.07)	1.50(0.04)	1.64(0.03)	2.46(0.05)	2.62(0.05)	2.75(0.05)	2.73(0.03)	2.61(0.02)
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)

In table 6, the efficiency of the proposed algorithm is compared with fourteen other algorithms including mQSO(5,5q) (Blackwell and Branke, 2006), mCPSO (Blackwell and Branke, 2006), AmQSO (Blackwell et al., 2008), mPSO (Kamosi et al., 2010), APSO (Rezazadeh et al., 2011), CLPSO (Hashemi and Meybodi, 2009a), FMSO (Li and Yang, 2008) , RPSO (Hu and Eberhart, 2002), SPSO(Du and Li, 2008), rSPSO(Bird and Li, 2007), PSO-CP(Liu, Yang and Wang, 2010), RVDEA(Woldesenbet and Yen, 2009), SFA(Nasiri and Meybodi, 2012) and CLDE(Noroozi, Hashemi and Meybodi, 2011) on MPB problems with various numbers of peaks, change frequency of 5000 and shift severity of 1. As it can be observed in this table, the proposed algorithm markedly outperforms other algorithms. In fact, the results show the proposed algorithm is robust with the number of peaks and whether the number of peaks is low or high, it leads to suitable results. Table 7 shows the results of the proposed method with different dimensions involving 10 peaks, change frequency of 5000 and shift severity of 1, in addition to those of mQSO, AmQSO, RPSO and mPSO. As can be seen, by increasing dimension of the problem space, the performance of the proposed method is less degraded, compared to other algorithms.

Table 7: Comparison between offline error (standard error) of the proposed method and that of 4 related algorithms on MPB with different dimensions, change frequency of 5000 and 10 peaks

	Shift Severity				
Algorithm	**2**	**5**	**10**	**15**	**20**
mQSO	1.01(0.04)	1.85(0.08)	4.22(0.20)	6.50(0.33)	8.88(0.34)
AmQSO	0.71(0.05)	1.51(0.10)	3.37(0.22)	4.91(0.31)	5.83(0.29)
mPSO	1.24(0.07)	1.61(0.12)	4.32(0.26)	7.07(0.25)	10.77(0.40)
RPSO	2.62(0.08)	12.98(0.48)	16.87(0.83)	18.48(0.97)	18.48(0.94)
PSO-AQ	**0.46(0.04)**	**0.89(0.03)**	**2.31(0.13)**	**4.01(0.32)**	**5.64(0.64)**

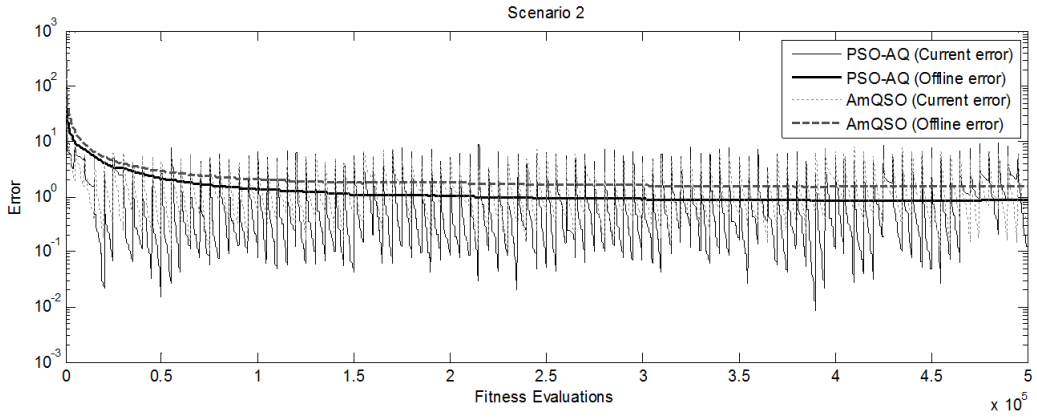

Figure 1: Offline error and current error curves of the proposed algorithm and AmQSO on Scenario 2 of MPB.

5 Conclusion

In this paper, a new method for optimization in dynamic environments was proposed based on particle swarm optimization algorithm. The proposed algorithm used a new quantum based local search for the sack of enhancing the algorithm efficiency. Results on various configurations of moving peak benchmark problem were compared with several other state of the art methods in this domain.

In the proposed algorithm, quantum particles approach has been employed for enhancing the swarm which was placed in the best peak of current environment. By decreasing the amount of quantums cloud regarding a novel self-adaptive mechanism, the ability of local search around the best peak was considerably increased. In fact, using this approach leaded to increasing speed for reaching the new position of optimum peak after each environment change. The experimental results showed that improving the ability of local search around best peak, significantly improved the algorithms efficiency. In addition, by increasing diversity

in swarms utilizing a novel method for randomized initializing of velocity vector components, other swarms could be quickly reached to new position of their goal without any additional computational cost.

Totally, in proposed algorithm, we attempted to satisfy all requirements of dynamic environments with MPB characteristics. Experimental results showed that the proposed algorithm has a suitable efficacy, compared to other algorithms in this domain.

Furthermore, there are some relevant works to pursue in the future. Controlling the value of fitness evaluation using the swarms placed in peaks which are not optimal leads to more time for reaching the better results to the swarms placed in the best peak. In addition, having two types with different configurations for finding and pursuing peak causes considerable improvement in algorithm's efficiency.

References

Andersen, H. 1991. *An investigation into genetic algorithms, and the relationship between speciation and the tracking of optima in dynamic functions*, PhD thesis, Honours thesis, Queensland university of technology, Brisbane, Australia.

Bird, S. and Li, X. 2007. Using regression to improve local convergence, *Proceedings of IEEE Congress on Evolutionary Computation(CEC 07)*, IEEE, Singapore, pp. 592–599.

Blackwell, T. and Branke, J. 2006. Multiswarms, exclusion, and anti-convergence in dynamic environments, *IEEE Transactions on Evolutionary Computation* **10**(4): 459–472.

Blackwell, T., Branke, J. and Li, X. 2008. Particle swarms for dynamic optimization problems, *Swarm Intelligence* pp. 193–217.

Branke, J. 1999. The moving peaks benchmark, *URL: http://www. aifb. uni-karlsruhe. de/ jbr/MovPeaks/movpeaks* .

Bui, L., Abbass, H. and Branke, J. 2005. Multiobjective optimization for dynamic environments, *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, Vol. 3, IEEE, pp. 2349–2356.

Carlisle, A. and Dozier, G. 2000. Adapting particle swarm optimization to dynamic environments, *Proceedings of the International Conference on Artificial Intelligence*, Athens, GA, USA: CSPIEA Press, pp. 429–434.

Cheng, H. and Yang, S. 2010. Multi-population genetic algorithms with immigrants scheme for dynamic shortest path routing problems in mobile ad hoc networks, *Applications of Evolutionary Computation* pp. 562–571.

Cobb, H. 1990. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments, *Technical report*, DTIC Document.

Du, W. and Li, B. 2008. Multi-strategy ensemble particle swarm optimization for dynamic optimization, *Information sciences* **178**(15): 3096–3109.

Grefenstette, J. et al. 1992. *Genetic algorithms for changing environments*, Navy Research Laboratory, Navy Center for Applied Research in Artificial Intelligence.

Hashemi, A. and Meybodi, M. 2009a. Cellular pso: A pso for dynamic environments, *Advances in Computation and Intelligence* pp. 422–433.

Hashemi, A. and Meybodi, M. 2009b. A multi-role cellular pso for dynamic environments, *Computer Conference, 2009. CSICC 2009. 14th International CSI*, IEEE, pp. 412–417.

Hu, X. and Eberhart, R. 2002. Adaptive particle swarm optimization: detection and response to dynamic systems, *Proceedings of the IEEE Congress on Evolutionary Computation(CEC 02)*, Vol. 2, IEEE, pp. 1666–1670.

Janson, S. and Middendorf, M. 2004. A hierarchical particle swarm optimizer for dynamic optimization problems, *Applications of evolutionary computing* pp. 513–524.

Jin, Y. and Branke, J. 2005. Evolutionary optimization in uncertain environments-a survey, *Evolutionary Computation, IEEE Transactions on* **9**(3): 303–317.

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

Kennedy, J. and Eberhart, R. 1995. Particle swarm optimization, *Neural Networks, 1995. Proceedings., IEEE International Conference on*, Vol. 4, IEEE, pp. 1942–1948.

Kennedy, J. and Mendes, R. 2002. Population structure and particle swarm performance, *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, Vol. 2, IEEE, pp. 1671–1676.

Li, C. and Yang, S. 2008. Fast multi-swarm optimization for dynamic optimization problems, *Fourth International Conference on Natural Computation*, IEEE, Jinan, China, pp. 624–628.

Li, C. and Yang, S. 2012. A general framework of multipopulation methods with clustering in undetectable dynamic environments, *Evolutionary Computation, IEEE Transactions on* **16**(4): 556–577.

Li, X., Branke, J. and Blackwell, T. 2006. Particle swarm with speciation and adaptation in a dynamic environment, *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, ACM, New York, NY, USA, pp. 51–58.

Liu, L., Yang, S. and Wang, D. 2010. Particle swarm optimization with composite particles in dynamic environments, *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **40**(6): 1634–1648.

Morrison, R. 2004. *Designing evolutionary algorithms for dynamic environments*, Springer.

Nanayakkara, T., Watanabe, K. and Izumi, K. 1999. Evolving in dynamic environments through adaptive chaotic mutation, *Third International Symposium on Artificial Life and Robotics*, Vol. 2, pp. 520–523.

Nasiri, B. and Meybodi, M. 2012. Speciation based firefly algorithm for optimization in dynamic environments, *International Journal of Artificial Intelligence* **8**(S12): 118–132.

Noroozi, V., Hashemi, A. and Meybodi, M. 2011. Cellularde: a cellular based differential evolution for dynamic optimization problems, *Adaptive and Natural Computing Algorithms* pp. 340–349.

Rezazadeh, I., Meybodi, M. and Naebi, A. 2011. Adaptive particle swarm optimization algorithm for dynamic environments, *Advances in Swarm Intelligence* pp. 120–129.

Vavak, F., Jukes, K., Fogarty, T. et al. 1998. Performance of a genetic algorithm with variable local search range relative to frequency of the environmental changes, *Genetic Programming* pp. 22–25.

Woldesenbet, Y. and Yen, G. 2009. Dynamic evolutionary algorithm with variable relocation, *Evolutionary Computation, IEEE Transactions on* **13**(3): 500–513.

Yang, S. and Yao, X. 2005. Experimental study on population-based incremental learning algorithms for dynamic optimization problems, *Soft Computing-A Fusion of Foundations, Methodologies and Applications* **9**(11): 815–834.

Yu, E. and Suganthan, P. 2009. Evolutionary programming with ensemble of explicit memories for dynamic optimization, *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, IEEE, pp. 431–438.