J. Basic. Appl. Sci. Res., 3(8)394-404, 2013

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

A New Particle Swarm Optimization Model based on Learning Automata Using Deluge Algorithm for Dynamic Environments

Mohammad Bager Moradi Geshlag[1], Jafar Sheykhzadeh[2]

[1]Department of Computer Engineering, Shabestar Branch, Islamic Azad University, Shabestar, Iran
[2]Department of Computer Engineering, Ahar Branch, Islamic Azad University, Ahar, Iran

ABSTRACT

Many optimization problems in the real world are dynamic in the sense that the global optimum value and the shape of fitness function may change with time. The task for the optimization algorithm in these environments is to find global optima quickly after the change in environment is detected. In this paper, we propose a new hybrid model of particle swarm optimization based on learning automata using Deluge algorithm which addresses this issue; this algorithm can provide a good diversity in the population of particles. In the proposed algorithm, for guidance of particles and their proper motivation and also following rapid changes in the environment, the combination of learning automata and Deluge algorithm is used in order to prevent particles to move toward the local optimums in the Search Space. Experimental results on different dynamic environments modeled by moving peaks benchmark show that the proposed algorithm outperforms other existing algorithms, for all tested environments.

KEYWORDS: Particle swarm optimization, Deluge algorithm, learning automata, dynamic environments, moving peaks function.

1. INTRODUCTION

Particle Swarm Optimization (PSO) is now established as an efficient optimization algorithm for static functions in a variety of contexts [18], [41], [42] and [43]. PSO is a population-based technique, similar in some respects to evolutionary algorithms (EAs), except that potential solutions (particles) move, rather than evolve, through the search space. The rules, or particle dynamics, which govern this movement, are inspired by models of swarming and flocking [2]. Each particle has a position and a velocity, and experiences linear spring-like [24] attractions toward two attractors:

1) the best position attained by that particle so far (particle attractor);
2) the best of the particle attractors in a certain neighborhood (neighborhood attractor);

Where best is in relation to evaluation of an objective function at that position. The swarm attractor therefore enables information sharing between particles, while the particle attractors serve as individual particle memories.

The optimization process is iterative. In each iteration, the acceleration vectors of all the particles are calculated based on the position of the corresponding attractors. Then, this acceleration is added to the velocity vector, the updated velocity is constricted so that the particles progressively slow down, and this new velocity is used to move the individual from the current to the new position. A more detailed introduction to PSO is provided in Section 2.1.

While most of the optimization problems discussed in the scientific literature are static, many real-world problems are dynamic, i.e., they change over time. In such cases, the optimization algorithm has to track a moving optimum as closely as possible, rather than just find a single good solution. It has been argued [29] that EAs may be a particularly suitable candidate for this type of problems, and over the past decade, a large number of EA variants for dynamic optimization problems have been proposed. For an overview, the reader is referred to [25], [26], [27] and [29].

Recently, the application of PSO to dynamic problems has also been explored [18], [19], [20], [21], [22], [23],and [28].Similar to EAs, PSO must be modified to not only find the optimal solution in a short time but also to be capable of tracking the solution after a change in the environment occurred. In order to have these capabilities, two important problems should be addressed for designing a particle swarm optimization algorithm for dynamic environments: outdated memory and diversity loss [6]. Outdated memory refers to the condition in which memory of the particles, that is the best location visited in the past and its corresponding fitness, may no longer be valid after a change in the environment [30]. Outdated memory problem is usually solved in one of these two ways: re-evaluating the memory [31] or forgetting the memory [20]. 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 loss 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 [32]. In the second approach, diversity is always monitored and as soon as it falls below a threshold, the swarm will be re-diversified.

[*]**Corresponding Author:** Mohammad Bager Moradi Geshlag, Department of Computer Engineering, Shabestar Branch, Islamic Azad University, Shabestar, Iran. E-mail: mb_moradi@iashab.ac.ir

In this paper we address diversity loss problem in adapting PSO to dynamic environments and propose a new method to solve it. To this aim, a combination of learning automata and Deluge algorithm are utilized to maintain diversity of particles and to scatter them over the search space. In the proposed algorithm, a Learning Automata is assigned to each particle and it controls the behavior of particle in the search space. Each learning automata has two main actions; "looking for the best experience" and "continuing the current path". In fact, "looking for the best experience" action selection will be followed by a local search and "continuing the current path "action selection cause global search and discovery unknown areas of the search space. Balancing between local and global search in the search process is the task of learning automata. When PSO algorithm will converge to a local optimum, the groups of particles lose the needed diversity to explore new areas and as a result the ability of algorithm is destroyed to react to environment changes. Deluge algorithm is used in order to create the needed diversity in the environment changes, which uses a dynamic mutation operation in the distance between two environment changes for this aim. Mutation operation caused a bigger search space area at the beginning of the search process and too smaller search space area at the end of search process (just before observing any change in the environment). Extensive experiments show that the proposed algorithm results less offline error than existing PSO model in literature, in environments which have many peaks or width and height of the peaks change very fast.

The rest of this paper is organized as follows; in section 2previous works reported in literature are reviewed. Section 3 introduces the proposed algorithm. Experimental results are presented in section 4 and section 5 is conclusion.

2. RELATED WORKS

2.1 Particle Swarm Optimization

The particle swarm optimization (PSO) algorithm is introduced by Kennedy and Eberhart [3] based on the social behavior metaphor.The fundament for the development of PSO is hypothesis that a potential solution to an optimization problem is treated as a bird without quality and volume, which is called a particle, flying through a D-dimensional space, adjusting its position in search space according to its own experience and its neighbors.

In PSO, Each particle has a component, called velocity that determines its route in the search space. PSO population includes all of the particles called the Swarm, and name of the Particle Swarm Optimization algorithm is derived from here. The beginning of PSO is like this; group of particles (solutions) are created randomly and updating over generations, they are trying to find optimal solution. At every step of the PSO, each particle is updated using two best values. First is the best position which particle is able to reach it. The mentioned position is recognized and maintained by pbest. Another best value used by algorithm is the best position obtained by the particle swarm since now. This position is displayed by gbest. After finding the best values, velocity and position of each particle are updated using Eqs.(1) and (2). In general, particle i of swarm, has position X_i^d and velocity V_i^d in the dimension d of the search space.

$$V_i^d(t+1) = w \times V_i^d(t) + c_1 \times r_1 \times \left(pbest_i^d(t) - X_i^d(t)\right) + c_2 \times r_2 \times \left(gbest(t) - X_i^d(t)\right) \text{Eqs(1)}$$

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1) \text{Eqs(2)}$$

In Eqs(1), $X_i = (X_i^1, X_i^2, \dots, X_i^d)$ is position and $V_i = (V_i^1, V_i^2, \dots, V_i^d)$ is velocity of particle i. the best position which is visited by particle is $pbest_i = (pbest_i^1, pbest_i^2, \dots, pbest_i^d)$ and the best position which is visited by the entire group is $gbest_i = (gbest_i^1, gbest_i^2, \dots, gbest_i^d)$. Random numbers r_1, r_2 are in the interval [0, 1]. $c_1, c_2$are cognitive component factor(the best solution which a particle can achieve to it lonely) and social component factor(the best solution that is diagnosed by the entire group) respectively.

Velocity of each particle in each dimension is limited toV_{max}. If the total acceleration caused the velocity in one dimension exceeds V_{max}, the value of velocity in that dimension is set toV_{max}. Right side of Eqs(1) is composed of three parts, first is the current velocity of particle, second and third parts are respectively responsible for speed change of particle and its rotation through the best personal experience and the best group experience. If the first part of this equation be ignored, then the velocity of particles is determined just respect to the current situation and the best experience of particle and the best experience of group. Thus, the best particle of group will remain in its place and others move toward that. In fact group motion of particles, without first part of Eqs(1) will be a process which causes the search space getting smaller and forming local search space around the best particle. On the other side, if only the first part of Eqs(1) be considered, particles move through their normal process until reach the limit and in the other word perform a global search [5]. In Eqs(1), combining these two idea, it is tried to balance local and global search.

In the above equation, parameter w is the inertia weight that is a better balance between local search and global search. For this purpose in this equation, this factor will be multiplied to the initial velocity of particle. It means that only some part of initial velocity is transmitted to the next velocity of the particle. Inertia weight can be a constant factor, a linear function of time or even a nonlinear function of time.

In PSO model existed particles in the problem space, try to move through the final solution, by setting their path and move to the best personal experience and best swarm experience.

Since in this algorithm, particles gradually go through the best founded solution so far, if this solution is a local optimum, all particles go through this local optimum and PSO algorithm does not provide any solution to go out of this local optimum. This is the biggest problem of PSO that causes disability in solving multi-peak problems especially with a big search space. When PSO algorithm converge to an optimum, particle swarm loose the required diversity to find new areas and in the result lose the ability of algorithm for showing reaction according to environment changes. The overall algorithm is summarized in figure (1).

Particle Swarm Optimization Algorithm
FOR EACH Particle i

Randomly initialize $\vec{v}_i, \vec{x}_i$ *and set* $\vec{pbest}_i = \vec{x}_i$

Evaluate $fitness(\vec{x}_i)$ *and* $fitness(\vec{pbest}_i)$

End for

$\vec{gbest} = \arg \max\{fitness(\vec{pbest}_i)\}$

REPEAT
FOR EACH Particle i
Calculate particle velocity according to Eqs. (1)

Update Particle position $\vec{x}_i$ According to Eqs. (2)

Evaluate $fitness(\vec{x}_i)$

//Update personal best

IF $fitness(\vec{x}_i) > fitness(\vec{pbest}_i)$ **THEN**

$\vec{pbest}_i = \vec{x}_i$

//Update global best

IF $fitness(\vec{x}_i) > fitness(\vec{gbest})$ **THEN**

$\vec{gbest} = \arg \max\{fitness(\vec{pbest}_i)\}$

End for
UNTIL termination criterion reached

Figure (1): The Pseudo-code of PSO

2.2 PSO in Dynamic Environment

The application of PSO to dynamic problems has been explored in various literatures. In this section we provide a brief overview of the most relevant works which addressing diversity loss in dynamic optimization problems.

Hu and Eberhart proposed re-randomization PSO for optimization in dynamic environments [32] in which some particles randomly are relocated after a change is detected or when the diversity is lost, to prevent losing the diversity. Li and Dam [33] showed that a grid-like neighborhood structure used in FGPSO [34] can perform better than RPSO in high dimensional dynamic environments by restricting the information sharing and preventing the convergence of particles to the global best position, thereby enhancing population diversity. Janson and Middendorf proposed HPSO, a tree-like structure hierarchical PSO [23], and reported improvements over standard PSO for dynamic environments. They also suggested Partitioned Hierarchical PSO in which a hierarchy of particles is partitioned into several sub-swarms for a limited number of generations after a change in the environment is detected [35]. Lung and Dumitresc [36] used two collaborating populations of equal size; one swarm is responsible for preserving the diversity of the particles by using a crowding differential evolutionary algorithm [37] while the other keeps track of global optimum with a PSO algorithm.

Blackwell and Bentley presented a repulsion mechanism in using the analogy of atom particles [21], [38]. In their model, a swarm is comprised of charged and neutral particles. The charged particles repel each other, leading to a cloud of charged particles orbiting a contracting, neutral, PSO nucleus. Moreover, Blackwell et al. extended the idea of charged particles to a quantum model and presented a multi-swarm method [6], [17], [22].

Du and Li [39] suggested an algorithm which divides particles into two parts. The first part uses a standard PSO enhanced by a Gaussian local search to find the global optimum quickly and the second part extends the searching area of the algorithm and patrols around the first part to track the changed global optimum which possibly escaped from the coverage of the first part. Although their algorithm performs well in the environments with one or two local optima, it cannot find the global optimum when the environment has more local optima.

Blackwell and Brank [6] suggested a method called MQSO that maintain variation in the two levels, this means that the particles are divided into several subgroups driven to different parts of the search space and each

group has a number of quantum particles that create diversity in group. The MQSO used a mechanism of excretion between the groups to prevent the some groups on the same local optimum. MQSO includes an additional mechanism named anti-convergence in order to identify new peaks and while all of the groups converge to the peaks where are located, enters the group with lower fitness into the total search space randomly. To particles in a group can pursue a mobile peak, diversity among particles within a group is required. Each group in MQSO includes a number of charged and quantum particles.

Li and Yang proposed a fast multi-swarm method (FMSO) which maintains the diversity through the run [7]. To meet this end two type of swarms are used: a parent swarm which maintains the diversity and detects the promising search area in the whole search space using a fast evolutionary programming algorithm, and a group of child swarms which explore the local area for the local optima found by the parent using a fast PSO algorithm. This mechanism makes the child swarms spread out over the highest multiple peaks, as many as possible, and guarantees to converge to a local optimum in a short time. Moreover, in [40], the authors introduced a clustering particle swarm optimizer in which a clustering algorithm partitions the swarm into several sub-swarms each searching for a local optimum.

Hashemi and Meybodi proposed a hybrid model of cellular automata and PSO [9], the main idea in this approach is using of local interactions in cellular automata and dividing the population of particles inside the cellular automata cells. Each group of particles trying to find a local optimum and it can cause to find global optimum. Some of multi-group methods which presented in papers to prevent existing more than one group on one peak, calculate distances between groups so two groups cannot work within other groups action radius and if this happens the group which has a lower fitness will get out of search space, in this method the computational cost has been deleted by using of cellular automata.

Kamosi, Hashemi and Meybodi proposed mPSO algorithm for dynamic environments [10], based on the multi-group and includes a number of parents and some of child which both groups exploit PSO technique to search. Parent groups are responsible for global search in the entire search space to find desired areas. When a desired area in the search space is found by parent group, a group of children in the area will be created to do local search in that area. Local search of children groups result in finding global optimums in the entire search space. Due to nature of the dynamic environment, it is likely that a local optimum convert to global optimum after a change in environment. As a result of maintaining information about local optimum caused a considerable increase of ability of the algorithm. One of the main advantages of this algorithm which caused better results comparing other methods is the ability of this algorithm in adjusting the number of existing groups in the environment appropriate to the number of its peaks.

In [11] an algorithm called HMSO is proposed for dynamic environments that is modified version of mPSO algorithm in [10]. HMSO algorithm, as the mPSO algorithm, includes a number of parent and children groups. HMPSO algorithm used the hibernation idea of animals to prevent useless search of child group and perform effective search. In this algorithm each child group is considered as an animal searching for food (meaning better solution). While a child group finds better solutions, it remains actively in the search space and continues to search. If a child group because of converging its particles to a solution which is not better than founded solutions by all other groups, being unable to find better solution, it means that the activity of than child group is useless. So until that child group can be useful diagnosing environment changes, it will be inactive. Inactive group is a group that exists in the search space, but does not do the search operation.

3. Optimization by using Learning Automata and Deluge Algorithm

PSO method cannot show appropriate response in dynamic environments. So changes must be made on PSO algorithm to offer an appropriate response. In this section, a hybrid technique based on PSO, using learning automata and Deluge algorithm is proposed for optimization in dynamic environments that can maintain diversity in the run time of algorithm. In the proposed algorithm, for conducting particles, their proper motions and also fast track of environmental changes, the combination of Learning Automata and Deluge Algorithm is used to prevent particles in the search space go toward local optimums and prepare needed diversity to explore in the search space. In the rest of this section, we have a brief overview of Learning Automata and Deluge algorithm, and then describe the proposed algorithm.

3.1 Learning Automata

Learning Automata are adaptive decision-making devices operating on unknown random environments. A Learning Automaton has a finite set of actions and each action has a certain probability (unknown to the automaton) of getting rewarded by the environment of the automaton. The aim is to learn to choose the optimal action (i.e. the action with the highest probability of being rewarded) through repeated interaction on the system. If the learning algorithm is chosen properly, then the iterative process of interacting on the environment can be made to result in selection of the optimal action. Figure (2) illustrates how a stochastic automaton works in feedback connection with a random environment. Learning Automata can be classified into two main families: fixed structure learning automata (FSLA) and variable structure learning automata (VSLA) [13], [14]. In the following, the variable structure learning automata which will be used in this paper is described.

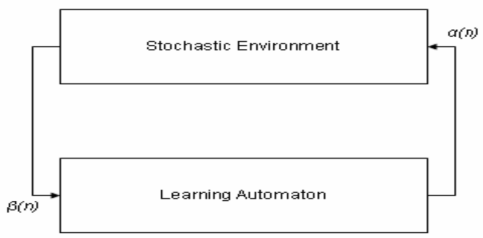

Figure (2): The interaction between learning automata and environment

A VSLA is a quintuple < α, β, p, T(α,β,p) >, where α, β, p are an action set with r actions, an environment response set and the probability set p containing r probabilities, each being the probability of performing every action in the current internal automaton state, respectively. If the response of the environment takes binary values learning automata model is P-model and if it takes finite output set with more than two elements that take values in the interval [0,1], such a model is referred to as Q-model, and when the output of the environment is a continuous variable in the interval [0,1], it is referred to as S-model. The function of T is the reinforcement algorithm, which modifies the action probability vector p with respect to the performed action and received response. Assume$\beta(i) \in [0,1]$. A general linear schema for updating action probabilities can be represented as follows. Let action i be performed then

$$p_j(n+1) = p_j(n) + \beta(n)[b/(r-1) - bp_j(n)] - [1-\beta(n)]ap_j(n) \qquad \forall j \quad j \neq i$$
$$p_j(n+1) = p_j(n) - \beta(n)bp_j(n) + [1-\beta(n)]a[1-p_j(n)] \text{Eqs(3)}$$

Where a and b are reward and penalty parameters. When a=b, the automaton is called L_{RP}. If b=0 the automaton is called L_{RI} and if $0 < b \ll a < 1$, the automaton is called $L_{R\varepsilon P}$. For more Information about learning automata the reader may refer to [13], [14].

3.2 Deluge Algorithm

Deluge Algorithm (DA) has been proposed by Dueck in [15]. In order to understand Deluge Algorithm better, consider the goal is finding the highest point in an area. Suppose that in the area, rain falls heavily. If the algorithm is considered as man, algorithm should move in this area In a way that it does not wet his feet. As the "water level" is enhancing, the algorithm can find the highest point. If we want to explain it algorithmically, this algorithm as well as other local search methods, replaces common solutions(s), if they are better than the best founded solutions(s*) so far. A new solution will be selected from previous neighborhood (N(s)). In Deluge algorithm solutions will be accepted which their fitness value is equal to or greater than water level (WL). WL value at each stage will increase (Up) with a specified size uniformly.

WL will continue to increase until it cannot find another answer better than the best discovered solution and WL is equal with the best found solution so far. At this stage, the algorithm will be repeated several times, and the algorithm terminates if a better result is not achieved. Initial value of WL is equal to initial obtained solution. Pseudo-code of Deluge algorithm is given in Figure (3).

```
Choose an initial configuration Old_Config
Choose WL0 and Up
For n=0 to number of iterations
    Generate a small stochastic perturbation New_Config of the solution
If Fitness (New_Config) > WL
Old_Config := New_Config
End If
    WL = WL + Up
End For
```

Figure (3): Deluge algorithm Pseudo-code

3.3Proposed Algorithm

In this section we proposed a new Particle Swarm Optimization model based on Learning Automata using Deluge Algorithm for optimization in dynamic environment that called LADABPSO[1]. In this algorithm a Learning Automata is assigned to each particle as the mastermind of the particle and it controls the behavior of particle in the search space. Each learning automata has two main actions; "looking for the best experience" and "continuing the current path". At first the position, velocity and action probability vector of learning automata will be initialized randomly. Until the maximum steps will be performed or the desired goal is achieved, the following steps are repeated:

1. Each learning automata chooses one of its actions according to its probability vector.

[1] Learning Automata and Deluge Algorithm Based Particle Swarm Optimization

2. Based on selected action by automata, type of updating the velocity is defined and the particle's position and velocity will be updated.
3. According to the results of particle's position updating, environment evaluates the selected action by learning automata and action selection probability vector of learning automata will be updated.

The chosen action in each step, determines the manner in which the particle velocity getting updated in that step. If the learning automata select "looking for the best experience" action, just looking for the personal and group best experience will be considered to update the particle velocity regardless of the current speed of the particle. Then the particle velocity can be updated according to Eqs(4).

$$V_i^d(t+1) = c_1 \times r_1 \times \left(pbest_i^d(t) - X_i^d(t)\right) + c_2 \times r_2 \times \left(gbest(t) - X_i^d(t)\right) \quad \text{Eqs(4)}$$

If the learning automata select "continuing the current path" action, the new particle velocity will be equal to the current speed of the particle and the particle will continue the current path. Then the particle position can be updated according to Eqs (2).

In fact, "looking for the best experience" action selection will be followed by a local search and "continuing the current path "action selection cause global search and discovery unknown areas of the search space. Balancing between local and global search in the search process is the task of learning automata. The chosen action evaluation method by learning automata is whether the new particle position is improved compared to previous position, learning automata are rewarded and otherwise it is fined. When PSO algorithm will converge to a local optimum, the groups of particles lose the needed diversity to explore new areas and as a result the ability of algorithm is destroyed to react to environment changes. Deluge algorithm is used in order to create the needed diversity in the environment changes, which uses a dynamic mutation operation in the distance between two environment changes as following; at first the probability value of mutation operation is equal to P_{max} and then gradually decreases its probability value until at the end of the change distance it reaches to the minimum value P_{min}. Mutation operation caused a bigger search space area at the beginning of the search process and too smaller search space area at the end of search process (just before observing any change in the environment).Particle coordinates in each dimension is mutated with probability $P_{mutation}$ independently. The value of particle coordinates is replaced with another value generated randomly with uniform distribution in an acceptable rage. The probability of mutation operation is selected during the algorithm execution from the interval $[P_{min}, P_{max}]$ dynamically. $P_{mutation}$ Parameter is set using Eqs (5).

$$P_{mutation} = \frac{(P_{max} - P_{min}) \times (interval - iter_{current} \; \mathbf{mod} \; interval)}{interval} + P_{min} \quad \text{Eqs(5)}$$

In Eqs(5), P_{max} and P_{min} are upper bound and lower bound of mutation operation probabilities respectively, interval parameter defines number of iterations between environment changes (environment changes frequency) and $iter_{current}$ is the current iteration number.When a change occurs in the environment, value of $P_{mutation}$ parameter is set to P_{max} and at the end of changes its value is set to P_{min}. The pseudo-code ofLADABPSO algorithm is shown in the figure (4).

The Pseudo-code of Learning Automata and Deluge Algorithm Based Particle Swarm Optimization

For each particle*i* **do**
Assign LA_i to particle i and Initialize the LA_i.
Initialize randomly $\vec{v}_i, \vec{x}_i$ *and set* $\overrightarrow{pbest}_i = \vec{x}_i$.
Evaluate $fitness(\vec{x}_i)$ *and* $fitness(\overrightarrow{pbest}_i)$.
Endfor

$$\overrightarrow{gbest} = \arg \max \{fitness(\overrightarrow{pbest}_i)\}$$

Repeat
Calculate $P_{mutation}$ according to Eqs(5).
if a change is detected in the environment **then**
Re-Initialize the particle.
Re-Evaluate the particle.
Update the $\overrightarrow{pbest}_i$ *and* $\overrightarrow{gbest}$.
Calculate $P_{mutation}$ according to Eqs(5).
Endif
For each particle i **do**
The LA_i selects an action α.
if α is "follow the best" **then**
Calculate particle velocity according to Eqs (4).
else
Set particle velocity to the previous velocity.
Endif
Update Particle position $\vec{x}_i$ according to Eqs(2).
Evaluate $fitness(\vec{x}_i)$
Update the probability vector of LA_i according to fitness of particle i.
//Update personal best
if $fitness(\vec{x}_i) > fitness(\overrightarrow{pbest}_i)$ **then**

$$\overrightarrow{pbest}_i = \vec{x}_i$$

Endif
//Update global best
if $fitness(\vec{x}_i) > fitness(\overrightarrow{gbest})$ **then**

$\overrightarrow{gbest} = \arg \max \{fitness(\overrightarrow{pbest_i})\}$
Endif
End for
// **Deluage Algorithm**
Choose WL and UP
Repeat
For each particle i **do**
Mutate the position of particle i with probability $P_{mutation}$
if $fitness\ of\ new_{position} > WL$ **then**
$\vec{x}_i = new_{position}$
Update the $\overrightarrow{pbest_i}\ and\ \overrightarrow{gbest}$
Endif
End for
$WL = WL + Up$
Until stop criterion of deluage algorithm is reached
Until termination criterion reached

Figure (4): LADABPSO Algorithm pseudo code

4. Experimental Study

In this section, we first describe moving peaks benchmark [16] on which the proposed algorithm 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. Dynamic Test Function

Branke in [16] introduced a dynamic benchmark problem, called moving peaks benchmark problem (Figure (5)); it is widely used as a benchmark to evaluate the performance of optimization algorithms in dynamic environments and it offers a reasonable simulation of optimization problem in the real world. Moving peaks function is proposed to create relationship between the complexity and the difficulty of understanding real world problems.

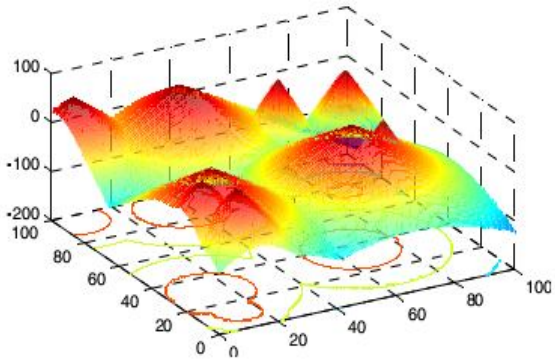

Figure (5): Moving peaks benchmark

Moving peak function includes *m* peaks in *n* dimension space with the parameters with real value. The fitness of environment is defined on the all of peak function as the maximum and it can be formulated as Eqs(6).

$$f(\vec{x}, t) = \max(B(\vec{x}), \max P(\vec{x}, h_i(t), w_i(t), \vec{p}_i(t))) \qquad i = 1 \dots m \qquad \text{Eqs(6)}$$

In Eqs (6), parameter $B(\vec{x})$is independent of time that specifies the basis of fitness value. Shape of peaks is determined by P. Each of variable peaks with time has height (*h*), width (*w*) and position (*p*). After a specified iterations (ΔE: environment changes frequency), the height, width and location of peak changes. The height and width of each peak are changed by adding random Gaussian variable and position of each peak will move by the vector V and the constant length S(number of peaks' movement in each frequency changes). Thus parameter S let control the intensity of a change. Parameter λ also determined that how the position changes of peak is related to its previous position. If λ=0 each movement is completely random and whenλ=1peak will always move in the same direction until it will osculate with space border and in this case it will bounce back to the environment like billiard balls. Generally parameter S allows us to control the intensity of changes and ΔE lets us to determine the frequency of changes and λ enables us to control movement route of changes. Relations that cause changes in a peak are as follows:

$$\sigma \in N(0,1)$$
$$h_i(t) = h_i(t-1) + height\_severity.\sigma$$
$$w_i(t) = w_i(t-1) + width\_severity.\sigma \qquad \text{Eqs(7)}$$
$$\vec{p}_i(t) = p_i(t-1) + \vec{v}_i(t)$$

In Eqs(7), parameters$h_i(t)$,$w_i(t)$and$\vec{p}_i(t)$are height, width and position vector of peak i at time t. Parameters *height_severity* and *width_severity* shows the severity of height changes and width of each peak,

respectively. Vector $\vec{v}_i(t)$(vector of peak*i* movement at time *t*) is calculated from Eqs(8) where the random vector $\vec{r}$is obtained by creating random numbers for each dimension and normalizing its length to S.

$$\vec{v}_i(t) = \frac{s}{|\vec{r} + \vec{v}_i(t-1)|}\left((1-\lambda)\vec{r} + \lambda\vec{v}_i(t-1)\right) \text{Eqs(8)}$$

Position, height and width of each peak are determined randomly with a set of limitations. Using different probability distributions, it is possible to produce many test problems with the same basic characteristics.

To measure the performance of algorithms, offline error average parameter is used. Offline error means the average fitness of the best founded positions by group of particles in each moment of time and the value of this parameter always is greater than/or equal to zero. If its value is closer to zero, it means better track the optimal solution simultaneous with the environment changes. The value of offline error is obtained by Eqs(9).

$$offline\_error = \frac{1}{T}\sum_{t=1}^{T}\left(fitness(swarm_{best}(t))\right) \text{Eqs(9)}$$

In Eqs(9),*T* is the maximum number of iterations (evaluations) and $\text{swarm}_{\text{best}}(t)$ is the best found position by the group of particles in iteration t. Default settings are listed in Table (1) for performed tests.

Table (1): Default settings for moving peaks benchmark

parameter	Value
Number of peaks	m = {1,5,10,20,30,40,50,100,200}
Frequency of changes	f = {500,1000,2500,5000,10000}
severity of changes in peaks height	7
severity of changes in peaks width	1
Shape of peaks	Conical (Cone)
Movement position amount of peaks	1
Search space dimension's	5
Range of peaks height	[30,70]
Range of peaks width	[1,12]
Standard height of peaks	50
Range of search space	$[0,100]^5$

4.2. Experimental Setting

For the proposed algorithm the number of particles are considered equal to 100, acceleration constants$c_1$and$c_2$equal to 1.496180,inertia weight (*w*) equal to 0.729844,$P_{max} = 0.6$and$P_{min} = 0.3$ and learning algorithm L_{RP} with values$\alpha = \beta = 0.01$have been used and the environment is interpreted as P-Model. In the tested experimentsfor proposed algorithm, the value of *WL* parameter is considered equal to the average fitness of all particles and the value of *Up* equal to *WL/5000*. The proposed algorithm is compared with FMSO[7], Adaptive mQSO[17], MQSO[6] , cellular PSO[9], mPSO[10] and HMSO[11] in different frequencies and different number of peaks based of average offline error criterion. For FMSO algorithm maximum number of child groups is 10 and excretion radius between child groups is 25 and the number of particles in parent groups and child groups are 100 and 10 respectively, this setting is proposed in [7]. $10(5+5^q)$ configuration is used in the experiments of MQSO where 10 groups are created and each group consists of 5 neutral particles and 5 quantum particles. Also for this algorithm quantum radius is equal to 0.5 and excretion and convergence radius is determined equally to 31.5. These settings are proposed in [6].Unlimited number of subgroups, which consisted of 5 neutral particles and 1 quantum particle, is considered for Adaptive mQSO algorithm. Five-dimensional cellular automata with 10^5 cells and Moore neighborhood with two cell radius in the search space is considered for Cellular PSO in experiments. The maximum particles velocity is equal to neighborhood radius and maximum number of particles is 10 per each cell and local search radius is determined to 0.5, also all of the particles run local search after observing change in the environment. These settings are proposed in [9].

The number of particles in parent and child groups is set to 5 and 10 respectively in mPSO algorithm. Also excretion radius between child groups and quantum particle radius is 30 and 0.5 respectively. Creation and excretion radius of groups are considered equal to 30 in the HmSO algorithm experiments and the number of active groups is set to 2 and the value of r_{conv} and the number of particles in each groups are 0.25 and 5 respectively.

4.3. Experiments results

For all algorithms we reported the average offline error and 95% confidence interval for 100 runs. to illustrate the response of algorithms to environmental changes, in every run of each algorithm, environment is changed 100 times with a certain frequency and each algorithm is run 100 times for each scenario and the average offline error is reported for this 100 times. Average Offline error of the proposed algorithm, FMSO [7], Adaptivem QSO[17], mQSO[6], cellular PSO[9], mPSO[10] and HmSO[11] for different dynamic environment is presented in tables 2 to 6. For each environment, result of the best performing algorithm(s) with 95% confidence is printed in bold.

In the fastest changing environment, where the peaks change every 500 iterations (Table 2), LADABPSO algorithm performs significantly better than other algorithms for different number of peaks. This is because the proposed algorithm quickly finds better solutions than other algorithms after a change occurs in the

environment, especially at the early iterations. This superiority also exist in the environment here the peaks change every 1000 iterations (Table 3), for different number of peaks but for the 10 peaks environment in which there is no significant difference between offline error for LADABPSO and Adaptive MQSO.

As depicted in table (4), where the frequency of environmental changes is 2500, the proposed algorithm outperforms other tested PSO algorithms for different number of peaks but for the 5 and 10 peaks environment, the result of proposed algorithm worse than other tested algorithm except FMSO.

By decreasing the dynamically of the environment (increasing f), a trend in difference of LADABPSO, Adaptivem QSO and HmSO can be seen in which LADABPSO breaks out to perform the same as or even worse than Adaptive mQSO and HmSO where the number of peaks is less than 30 (Table 5and 6).

Table (2): offline error average for different number of peaks in frequency 500

Number of peaks	FMSO	Adaptive mQSO	mQSO	CellularPSO	mPSO	HmSO	Proposed Algorithm
1	27.58±0.09	5.08±0.27	36.52±3.2	22.37±3.8	8.71±0.48	8.53±0.49	3.68±0.23
5	19.45±0.4	5.14±0.09	13.50±0.8	14.20±0.6	6.69±0.26	7.40±0.31	5.00±0.14
10	18.26±0.3	6.20±0.11	11.18±0.4	13.55±0.5	7.19±0.23	7.56±0.27	6.19±0.45
20	17.34±0.3	6.94±0.18	10.54±0.2	12.77±0.3	8.01±0.19	7.81±0.20	6.91±0.45
30	16.39±0.4	7.23±0.16	10.37±0.2	12.55±0.4	8.43±0.17	8.33±0.18	5.48±0.08
40	15.34±0.4	7.43±0.17	10.32±0.2	12.33±0.3	8.62±0.18	8.45±0.18	5.14±0.16
50	15.54±0.2	7.49±0.09	10.33±0.2	12.19±0.3	8.76±0.18	8.83±0.17	4.69±0.25
100	12.87±0.6	7.29±0.15	9.93±0.21	11.38±0.2	8.91±0.17	8.85±0.16	4.12±0.15
200	11.52±0.6	6.82±0.14	9.67±0.20	11.34±0.2	8.88±0.14	8.85±0.16	3.28±0.10

Table (3): offline error average for different number of peaks in frequency 1000

Number of peaks	FMSO	Adaptive mQSO	mQSO	CellularPSO	mPSO	HmSO	Proposed Algorithm
1	14.42±0.4	2.68±0.14	19.1±1.5	7.97±0.54	4.44±0.24	4.46±0.26	2.04±0.08
5	10.59±0.2	3.22±0.07	7.59±0.39	6.16±0.31	3.93±0.16	4.27±0.08	3.16±0.13
10	10.40±0.1	4.11±0.08	6.33±0.23	5.94±0.23	4.57±0.18	4.61±0.07	4.13±0.15
20	10.33±0.1	4.75±0.14	6.46±0.20	6.13±0.17	4.97±0.13	4.66±0.12	4.24±0.03
30	10.06±0.1	4.98±0.10	6.51±0.16	6.23±0.15	5.15±0.12	4.83±0.09	4.25±0.13
40	9.85±0.1	5.10±0.11	6.43±0.18	6.27±0.15	5.17±0.10	4.82±0.09	4.55±0.04
50	9.54±0.1	5.12±0.05	6.67±0.16	6.26±0.16	5.33±0.10	4.96±0.03	3.73±0.13
100	8.77±0.0	5.03±0.09	6.34±0.12	6.27±0.14	5.60±0.09	5.14±0.08	3.19±0.18
200	8.06±0.0	4.65±0.09	6.13±0.12	6.01±0.11	5.78±0.09	5.25±0.08	2.88±0.03

Table (4): offline error average for different number of peaks in frequency 2500

Number of peaks	FMSO	Adaptive mQSO	mQSO	CellularPSO	mPSO	HmSO	Proposed Algorithm
1	6.29±0.20	1.09±0.06	7.79±0.72	4.57±0.31	1.79±0.10	1.75±0.10	0.87±0.01
5	5.03±0.12	1.58±0.13	3.53±0.18	3.15±0.21	2.04±0.12	1.92±0.11	4.27±0.18
10	5.09±0.09	2.33±0.11	3.20±0.14	3.09±0.16	2.66±0.16	2.39±0.16	4.23±0.11
20	5.32±0.08	2.84±0.09	3.83±0.12	3.60±0.13	3.07±0.11	2.46±0.09	2.32±0.15
30	5.22±0.08	3.13±0.09	4.03±0.12	3.88±0.12	3.15±0.08	2.57±0.05	2.55±0.04
40	5.09±0.06	3.23±0.08	3.90±0.11	4.17±0.12	3.17±0.07	2.56±0.06	2.13±0.01
50	4.99±0.06	3.24±0.07	3.95±0.10	4.25±0.12	3.26±0.07	2.65±0.05	2.18±0.23
100	4.60±0.05	3.20±0.06	3.81±0.10	4.25±0.13	3.31±0.05	2.72±0.04	2.14±0.17
200	4.34±0.04	3.00±0.05	3.66±0.07	4.20±0.09	3.36±0.05	2.81±0.04	2.16±0.28

Table (5): offline error average for different number of peaks in frequency 5000

Number of peaks	FMSO	Adaptive mQSO	mQSO	CellularPSO	mPSO	HmSO	Proposed Algorithm
1	3.44±0.11	0.55±0.02	4.06±0.40	2.79±0.19	0.90±0.05	0.87±0.05	0.88±0.31
5	2.94±0.07	1.00±0.04	1.98±0.10	1.94±0.18	1.21±0.12	1.18±0.04	1.82±0.34
10	3.11±0.06	1.43±0.04	1.93±0.09	1.93±0.08	1.61±0.12	1.42±0.04	1.86±0.51
20	3.36±0.06	1.95±0.05	2.59±0.11	2.73±0.12	2.05±0.08	1.50±0.06	1.61±0.13
30	3.28±0.05	2.15±0.05	2.84±0.08	3.08±0.11	2.18±0.06	1.65±0.04	1.37±0.06
40	3.26±0.04	2.28±0.04	2.74±0.08	3.28±0.11	2.24±0.06	1.65±0.05	1.09±0.12
50	3.22±0.05	2.28±0.02	2.74±0.07	3.34±0.07	2.34±0.06	1.66±0.02	1.05±0.03
100	3.06±0.04	2.31±0.03	2.61±0.06	3.48±0.11	2.32±0.04	1.68±0.03	1.07±0.14
200	2.84±0.03	2.11±0.03	2.45±0.05	3.37±0.08	2.34±0.03	1.71±0.02	1.12±0.24

Table (6): offline error average for different number of peaks in frequency 10000

Number of peaks	FMSO	Adaptive mQSO	mQSO	CellularPSO	mPSO	HmSO	Proposed Algorithm
1	1.90±0.06	0.27±0.01	2.24±0.19	1.63±0.12	0.44±0.02	0.45±0.02	0.22±0.01
5	1.75±0.06	0.54±0.09	1.07±0.07	1.20±0.15	0.72±0.08	0.71±0.12	0.56±0.01
10	1.91±0.04	0.86±0.04	1.19±0.09	1.19±0.09	1.05±0.10	0.94±0.09	0.65±0.02
20	2.16±0.04	1.24±0.06	1.86±0.09	2.13±0.10	1.37±0.06	1.02±0.06	1.08±0.14
30	2.18±0.04	1.43±0.05	2.09±0.08	2.59±0.12	1.50±0.06	1.13±0.04	1.12±0.02
40	2.21±0.03	1.57±0.05	2.02±0.06	2.72±0.09	1.53±0.05	1.09±0.04	1.14±0.01
50	2.60±0.08	1.58±0.02	2.03±0.07	2.91±0.11	1.62±0.04	1.10±0.03	1.09±0.03
100	2.20±0.03	1.65±0.04	1.94±0.05	3.02±0.11	1.62±0.03	1.08±0.02	1.06±0.02
200	2.00±0.02	1.50±0.03	1.79±0.04	2.94±0.10	1.64±0.02	1.07±0.02	1.07±0.01

5. CONCLUSION

In this paper, a new model of particle swarm optimization is proposed based on Learning Automata using Deluge Algorithm in dynamic environment. In the proposed model a Learning Automata is dedicated to each particle in the group which its task is controlling the behavior and motion of particle. Deluge algorithm is used in the proposed model for better diversification among environmental changes, which uses a dynamic mutation operation in the environmental changes.

Compared to other well-known approaches, our proposed model results more accurate solutions in highly dynamic environments, modeled by MPB[16], where peaks change in position, width and height.

REFERENCES

[1] N.Raman and F.B., “Talbot the job shop tardiness problem: a decomposition approach”, European journal of apportional Vol.69, pp.187-199, 1993.

[2]Kennedy, J., Eberhart, R.C., “Particle Swarm Optimization”, IEEE International Conference on Neural Networks, Piscataway, NJ, vol. IV, pp. 1942–1948 (1995).

[3] Y.Chunming , D. Simon ,“A new particle swarm optimization technique”, 18th International Conferences System Engineering, 2 .ICSEng 2005,pp.164-169.

[4]J.Kennedy and R.C.Eberhart,”A Discrete Binary Version of the Particle Swarm Algorithm”, 1997 IEEE International Conference on Computational Cybernetics and Simulation, Vol.5, pp.4104-4108.

[5] Shi, Y. and Eberhart, R. C., “A Modified Particle Swarm Optimizer”, IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA, 1998.

[6] T. M. Blackwell and J. Branke,“Multi-swarms, exclusion and anti-convergencein dynamic environments”, IEEE Transactions on Evolutionary Computation, Vol.10, pp.459–472, 2006.

[7]Li, C., Yang, S., “Fast Multi-Swarm Optimization for Dynamic Optimization Problems”, Fourth International Conference on Natural Computation, Jinan, Shandong, China, vol. 7, pp.624–628, 2008.

[8] X. Yao and Y. Liu., “Fast evolutionary programming”, Proc.Of the 5th Annual Conference on EvolutionaryProgramming, pp.451-460, 1996.

[9] B. Hashemi and M. R. Meybodi, “Cellular PSO: A PSO for Dynamic Environments”, in Advances in Computation and Intelligence, Lecture Notes in Computer Science, vol. 5821, pp. 422-433, 2009.

[10] M. Kamosi,A. B.Hashemi, M. R.MeybodiM. R., “A New Particle Swarm Optimization Algorithm for Dynamic Environments”, LNCS 6466, pp. 129-138, 2010.

[11] M. Kamosi,A. B.Hashemi, M. R.Meybodi M. R., “A Hibernating Multi-Swarm Optimization Algorithm for Dynamic Environments”, Nature and Biologically Inspired Computing (NaBIC),2010.

[12] C. Li, Y. Liu, L. Kang and AM. Zhou,“A Fast ParticleSwarm Optimization Algorithm with CauchyMutation and Natural Selection Strategy”. Springer, LNCS4683, pp.334– 343, 2007.

[13] K. Narendra and M. A. L. Thathachar, “Learning Automata: An Introduction”, Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[14] K. Najim and A. S. Poznyak, “Learning Automata: Theory and Application”, Tarrytown, NY: Elsevier Science Ltd., 1994.

[15] Dueck G., Gunter, “New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel”, Journal of Computational Physics, Vol. 104, pp. 86-92, 1993.

[16] J. Branke, “Memory Enhanced Evolutionary Algorithms for Changing Optimization Problems”, in 1999 Congress on Evolutionary Computation, Washington D.C., USA, 1999, pp. 1875-1882.
[17]T. Blackwell, J. Branke, and X. Li, "Particle Swarms for dynamic Optimization Problems”, in Swarm Intelligence, Natural Computing Series, vol. Part II, 2008, pp. 193-217.
[18] K. Parsopoulos and M. Vrahatis, “Recent approaches to global optimization problems through particle swarm optimization” , Natural Comput.,vol. 1, no. 2–3, pp. 235–306, 2002.
[19]X. Hu and R. Eberhart, “Adaptive particle swarm optimization: Detec-tion and response to dynamic systems”, inProc. Congr. Evol.Comput, 2002, pp. 1666–1670.
[20] A. Carlisle and G. Dozier, “Adapting particle swarm optimization to dynamic environments”, in Proc. Int. Conf. Artif.Intell.2000, pp.429–434.
[21] T. Blackwell,“Swarms in dynamic environments”, inProc. Genetic and Evol.Comput. Conf., vol. 2723, E. Cantu-Paz, Ed., 2003, pp. 1–12.
[22] T. Blackwell and J. Branke,“Multi-swarm optimization in dynamic environments”, in Applications of Evolutionary Computing. Ser. Lecture Notes in Computer Science, G. R. Raidlet al., Eds. Berlin, Germany: Springer-Verlag, 2004, vol. 3005, pp. 489–500.
[23] S. Janson and M. Middendorf,“A hierarchical particle swarm optimizer for dynamic optimization problems”, in Applications of Evolutionary Computing. Ser. Lecture Notes in Computer Science, G. R. Raidl,Ed. Berlin, Germany: Springer-Verlag, 2004, vol. 3005, pp. 513–524.
[24] B. Brandstätter and U. Baumgartner, “Particle swarm optimization-Mass spring system analog” , IEEE Trans. Magn., vol. 38, no.2, pp. 997–1000, 2002.
[25] J. Branke and H. Schmeck, “Designing evolutionary algorithms for dynamic optimization problems”, in Theory and Application of Evolutionary Computation: Recent Trends, S. Tsutsui and A. Ghosh, Eds. Berlin, Germany: Springer-Verlag, 2002, pp. 239–262.
[26] R. Morrison, “Designing Evolutionary Algorithms for Dynamic Environments”, Berlin, Germany: Springer-Verlag, 2004.
[27]Y. Jin and J. Branke, “Evolutionary optimization in uncertain environments: a survey”, IEEE Trans. Evol. Comput., vol. 9, no. 3, pp.303–317, Jun. 2005.
[28] X. Zhang, L. Yu, Y. Zheng, Y. Shen, G. Zhou, L. Chen, L. Xi, T. Yuan,J. Zhang, and B. Yang, “Two-stage adaptive PMD compensation in a 10 Gbit/s optical communication system using particle swarm optimization algorithm” , Optics Commun., vol. 231, pp. 233–242, 2004.
[29] --, “Evolutionary Optimization in Dynamic Environments”, Norwell, MA: Kluwer, 2001.
[30] Blackwell.T, “Particle swarm optimization in dynamic environments” , in: Evolutionary Computation in Dynamic and Uncertain Environments, pp. 29–49 (2007).
[31] Eberhart, R.C., Shi, Y., “Tracking and optimizing dynamic systems with particle swarms” ,In: IEEE Congress on Evolutionary Computation, Seoul, Korea, vol. 1, pp. 94–100 (2001).
[32] Hu, X., Eberhart, R.C.,“Adaptive particle swarm optimization: detection and response to dynamic systems” ,In: IEEE Congress on Evolutionary Computation, Honolulu, HI, USA, vol. 2, pp. 1666–1670 (2002).
[33] Li, X., Dam, K.H.,“Comparing particle swarms for tracking extrema in dynamic environments” ,In: IEEE Congress on Evolutionary Computation, Canberra, Australia, pp. 1772–1779 (2003).
[34] Kennedy, J., Mendes, R., “Population structure and particle swarm performance”, in: Evolutionary Computation Congress, Honolulu, Hawaii, USA, pp. 1671–1676 (2002).
[35] Janson, S., Middendorf, M., “A hierarchical particle swarm optimizer for noisy anddynamic environments”,Genetic Programming and Evolvable Machines 7, 329–354 (2006).
[36] Lung, R.I., Dumitrescu, D., “A collaborative model for tracking optima in dynamicenvironments”, in: IEEE Congress on Evolutionary Computation, Singapore, pp. 564–567(2007).
[37] Thomsen, R., “Multimodal optimization using crowding-based differential evolution”, In:IEEE Congress on Evolutionary Computation, Portland, Oregon, USA, pp. 1382–1389 (2004).
[38] Blackwell, T.M., Bentley, P.J., “Dynamic search with charged swarms”, In: Genetic andevolutionary computation conference, pp. 19–26. Morgan Kaufmann Publishers Inc., New York (2002).
[39] Du, W., Li, B., “Multi-strategy ensemble particle swarm optimization for dynamicoptimization", Information Sciences 178, 3096–3109 (2008).
[40] Li, C., Yang, S., “A clustering particle swarm optimizer for dynamic optimization”, In: IEEECongress on Evolutionary Computation, pp. 439–446 (2009).
[41] S.Ghatei, F. TighPanahi, M. Hosseinzadeh, M. Rouhi, I. Rezazadeh, A .Naebi,Z. Ghatei and R. Pasha Khajei,“A New Hybrid Algorithm for Optimization Using PSO and GDA”, Journal of Basic and Applied Scientific Research,2(3)2336-2341,2012.
[42] A.hazrati, S. khezri and S. Ghatei,“Compare between CLA-EC and PSO-Great Deluge Mechanism as Feature Selection in Facial Expression Recognition System”, Journal of Basic and Applied Scientific Research,3(3)130-136,2013.
[43] M. Abedini, H. Saremi,“A Hybrid of GA and PSO for Optimal DG Location and Sizing in Distribution System with Load Uncertainty”, Journal of Basic and Applied Scientific Research, 2(5)5103-5118, 2012.