New Firefly Algorithm based On Multi swarm & Learning Automata in Dynamic Environments

Azam Amin Abshouri
Department of Computer Engineering, Islamic Azad University, Qazvin branch, Iran
Tehran, Iran
e-mail: a_amin1980@yahoo.com

Mohammad Reza Meybodi
Department of Computer & IT Engineering, Amirkabir University of Technology, Tehran, Iran
Tehran, Iran
e-mail: mmeybodi@aut.ac.ir

Alireza Bakhtiary
Department of science, university of Tehran, Tehran, Iran
Tehran, Iran
e-mail: bakhtiaryalireza@yahoo.com

***Abstract*——Many real world problems are mostly time varying optimization problems, which require special mechanisms to detect changes in environment and then response to them. In this paper, combination of learning automata and firefly algorithm in order to improve the performance of firefly algorithm in dynamic environments has been proposed. In the algorithm, the firefly algorithm has been equipped with three learning automatons and velocity parameter, so they can increase diversity in response the dynamic environments. The main idea is based to split the population of fireflies into a set of interacting swarms. This Algorithm evaluated on a variety of instances of the multimodal dynamic moving peaks benchmark. Results are also compared with alternative approaches from the literature. They show that the new algorithm optimizer significantly outperforms previous approaches**.

Keywords-component; Dynamic Environment; Learning Automata; Multi Swarm; optimization methods & firefly Algorithm

I. INTRODUCTION

In general, optimization algorithms can be classified into two main categories: deterministic and stochastic. Deterministic algorithms such as hill-climbing will produce the same set of solutions if the iterations start with the same initial guess. On the other hand, stochastic algorithms often produce different solutions even with the same initial starting point. However, the final results, though slightly different, will usually converge to the same optimal solutions within a given accuracy.

Most stochastic algorithms can be considered as metaheuristic, and good examples are genetic algorithms (GA) and particle swarm optimization (PSO). Many modern metaheuristic algorithms were developed based on the swarm intelligence in nature. New modern metaheuristic algorithms are being developed and begin to show their power and efficiency. For example, the Firefly Algorithm developed by the author shows its superiority over some traditional algorithms [1].

Most of the real world problems are known as optimization problems. In these problems, the extrema change during the time over the landscape [2]. Due to the lack of traditional optimization algorithms to trace the optima, they failed in the non stationary environments. Therefore the bio-inspired algorithms are more frequently used in order to solve different optimization problems such as dynamic optimization problems [3]. In dynamic optimizations, the algorithms not only have to detect the changes but also they must response to the changes in the landscape. There are several suggestions to deal with time varying optimization have been proposed based on population based algorithms in [4-5].

The multiswarm idea has also been proposed for purposes other than tracking dynamic environments. For example, the nichePSO approach of Brits et al. [6] is aimed at multimodal functions. It creates a two particle subswarm from a particle and its nearest spatial neighbor, if the variance in that particles fitness is less than a threshold. Subswarms may merge, and they also absorb particles from the main swarm. Although nichePSO was able to optimize successfully some static multimodal benchmark functions, it is not adaptable to the dynamic problem in an obvious way. This is because the algorithm, as the authors point out, depends on a very uniform distribution of particles in the search space, and on a training phase. The already mentioned multiswarm approach with clearing has also been used in [7] for detecting several optima in multimodal environments. In [8], different swarms are used to optimize different parts of a solution cooperatively. A two swarm approach for min–max optimization was proposed in [9]. Kennedy [10] suggested to cluster particles in each iteration into subswarms, and have them determine their global best separately. In [11], multiswarm variants have also been suggested in a nonoptimization context. Finally, what was proposed In [12] is based on PSO and split the population of particles into a set of interacting swarms. These swarms interact locally by an exclusion parameter and globally through a new anti-convergence operator. In addition, each swarm maintains diversity either by using charged or quantum particles.

978-1-4577-0174-0/11/$26.00 ©2011 IEEE

Learning automata are adaptive decision-making devices that operating in an unknown random environment and progressively improve their performance via a learning process. It has been used successfully in many applications such as call admission control in cellular networks [13] and [14], capacity assignment problems [15], Adaptation of back propagation parameter [16], and Determination of the number of hidden units for three layers neural networks [17]. In this paper, we propose a new approach for adaptive adjusting parameters of firefly, absorption Coefficient and randomization Coefficient. In the proposed approach, parameters of firefly are set by means of two learning automata, one learning automata for each parameter. Results and tests have shown the standard functions. The algorithm works better than firefly algorithm standard and PSO and its improved algorithms [18].

The proposed algorithm is based on firefly, Learning Automata, multiswarm, exclusion and anti-convergence. Since for any firefly In standard firefly algorithm there are three parameter that are fixed from beginning to end, in this Algorithm, we equip any firefly with three learning automaton to determine these parameters. Further, another factor effective in the movements of fireflies is velocity. This factor is also added in order to improve the step size.

In the rest of the paper the following materials are provided. Section 2 reviews the firefly Algorithm. Section 3 gives a brief introduction to learning automata. The proposed algorithm is given in section 4. Experiments settings and results are presented in section 5. Section 6 concludes the paper.

II. FIREFLY ALGORITHM

The flashing light of fireflies is an amazing sight in the summer sky in the tropical and temperate regions. There are about two thousand firefly species, and most fireflies produce short and rhythmic flashes. The pattern of flashes is often unique for a particular species. The flashing light is produced by a process of bioluminescence, and the true functions of such signaling systems are still debating. However, two fundamental functions of such flashes are to attract mating partners (communication), and to attract potential prey. In addition, flashing may also serve as a protective warning mechanism. The rhythmic flash, the rate of flashing and the amount of time form part of the signal system that brings both sexes together. Females respond to a male's unique pattern of flashing in the same species, while in some species such as photuris, female fireflies can mimic the mating flashing pattern of other species so as to lure and eat the male fireflies who may mistake the flashes as a potential suitable mate.[19]

The flashing light formulated by Yang and it was based on the idealized behavior of the flashing characteristics of fireflies. For simplicity, in describing our Firefly Algorithm (FA), we now use the following three idealized rules:

1) All fireflies are unisex so that one firefly will be attracted to other fireflies regardless of their sex;

2) Attractiveness is proportional to their brightness, thus for any two flashing fireflies, the less bright one will move towards the brighter one. The attractiveness is proportional to the brightness and they both decrease as their distance increases. If there is no brighter one than a particular firefly, it will move randomly;

3) The brightness of a firefly is affected or determined by the landscape of the objective function.[1]

In this algorithm, Assume that there exists a swarm of m agents(fireflies) solving above mentioned problem iteratively and xi represents a solution for a firefly i in algorithm's iteration k, whereas f(xi) denotes its cost. Initially all fireflies are dislocated in S (randomly or employing some deterministic strategy). Each firefly has its distinctive attractiveness which implies how strong it attracts other members of the swarm. As a firefly attractiveness one should select any monotonically decreasing function of the distance rj = d(xi; xj) to the chosen firefly j, e.g. the exponential function:

$$\beta=\beta_0\times e^{-\gamma r_{ij}^2} \qquad (1)$$

Where β_0 and γ are predetermined algorithm parameters: maximum attractiveness value and absorption coefficient, respectively. Furthermore every member of the swarm is characterized by its light intensity Ii which can be directly ex pressed as a inverse of a cost function $f(x_i)$. To effectively explore considered search space S it is assumed that each firefly *i* is changing its position iteratively taking into account two factors: attractiveness of other swarm members with higher light intensity i.e. $I_j > I_i$; $\forall j = 1..m; j \neq i$ which is varying across distance and a fixed random step vector u_i. It should be noted as well that if no brighter firefly can be found only such randomized step is being used. Finally Pseudo-cod's firefly algorithm is presented in *"Fig. 1"*[30].

```
Objective function f(x), x=(x1,x2,…,xd)T
Initialize a population of fireflies
Define light absorption coefficient γ
While (t<MaxGeneration)
 For i=1:n (all n fireflies)
  For j=1:i
  Light intensity  is determined by f(xi)
    If (Ij>Ii)
     Move firefly i towards j in all d dimensions
    End if
    Evaluate new solutions and update light intensit
    End for j
 End for i
  Rank the fireflies and find the current best
End while
```

Figure 1. Pseudo-cod's Firefly Algorithm

III. LEARNING AUTOMATA

Learning Automata are adaptive decision-making devices operating on unknown random environments [20]. The Learning Automaton has a finite set of actions and each action has a certain probability (unknown for 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 select the optimal action.

Figure 1 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 and variable structure learning automata (VSLA). In the following, the variable structure learning automata is described.

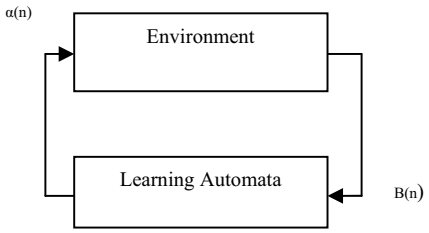

Figure 2- The interaction between learning automata and environment

Variable structure learning automata can be shown by a quadruple { α , β, p, T } where $\alpha=\{\alpha_1, \alpha_2, ..., \alpha_r\}$ which is the set of actions of the automaton, $\beta=\{\beta_1, \beta_2,\ldots, \beta_m\}$ is its set of inputs, p={p1, ..., pr} is probability vector for selection of each action, and p(n +1) = T[α (n),β(n), p(n)] is the learning algorithm. If β={0,1}, then the environment is called P-Model. If β belongs to a finite set with more than two values, between 0 and 1, the environment is called Q-Model and if β is a continuous random variable in the range [0, 1] the environment is called S-Model. Let a VSLA operate in an SModel environment. A general linear schema for updating action probabilities when action i is performed is given by:

$$P_i(n+1) = P_j(n) + a.(1- p_i(n)) - (b.\beta_i(n).pi(n)$$
$$P_j(n+1) = P_j(n) + a.(1- \beta_i(n)).Pj(n) + (b.\beta_i(n))$$
$$[1/(r-1) - p_j(n)] \quad \forall j \; j \neq i \qquad (2)$$

Where a and b are reward and penalty parameters. When a=b, the automaton is called S-LR-P. If b=0 and 0<b<<a<1, the automaton is called S-LR-I and S-LR P, respectively [21].

IV. PROPOSED ALGORITHM

The fundamental idea of the proposed approach is to divide up the swarm into a number of subswarms, in order to position each of those subswarms on different, promising peaks of the landscape. This is undesirable since the motivation behind a multiswarm approach is to position different swarms on different peaks. Should one of the watched peaks become optimal, a swarm will already be presented to take advantage[12]. In this paper, two forms of swarm interaction are applied: exclusion and anti-convergence. Exclusion is a local interaction between colliding swarms, preventing swarms from settling on the same peak. Anti-convergence is an information sharing interaction among all swarms in the multiswarm algorithm, with the aim of allowing new peaks to be detected. If the number of swarms is less than the number of peaks in the fitness landscape, and all swarms are converged (to different peaks, because exclusion is in force), the system will lose its peak-detection capability, then the swarm with the worst function value will reinitialized in the search space. Also, if distance two swarms is less than r_{excl}, then the swarm with the worst function value is reinitialized in the search space.[12]

Further, the proposed algorithm is based on firefly movement in standard firefly algorithm. In standard firefly algorithm, for any two flashing fireflies, the less bright one will move towards the brighter, which includes a percentage of randomness. The initial percentage value and other parameter such as attractiveness Coefficient is crucially important in determining the speed of the convergence and how the FA algorithm behaves. In mode standard, the parameters are fixed from beginning to end. Alpha determines random percentage in firefly moving. It includes value between zero to one. Absorption coefficient is named gamma. The parameter varies between zero to extreme. If Coefficient is close to zero, then $\beta = \beta_0$ and this corresponds to a special case of particle swarm optimization. Besides, if absorption coefficient is close extremely, this is the case where the fireflies fly in a very foggy region randomly. Finally β_0 is maximum Attractiveness value. Therefore in the algorithm, any firefly equipped with three learning automatons; the first learning automata represents absorption Coefficient, the second learning automata represents maximum attractiveness value and the last learning automata represents randomization coefficient.

All of the LAs are P-Model. Initial values for α in all LA are set according to defined range. Initial value of α in the first and second LA may change according to [9]; it means, in each iteration, the values of α are set as:

I. is unchanged;
II. is increased by multiplying a number greater than one;
III. is decreased by multiplying a number smaller than one. This number is generated Based on the current repetition number by "(3)",

$$Witr = X + \frac{itr_{max} - itr}{itr_{max}} \times (Y-X) \qquad (3)$$

In "(3)", n is producer nonlinear index, in this formula, in any iteration, weight decreases nonlinear from Y to X.

It should be mentioned, during steps adapt their environment, we will not let values of α change in unauthorized range.

The initial values for α in the last LA are set as follows:

1- we set zero value;
2- we set negative value;
3- we set positive value.

In addition, we add the velocity parameter of PSO to every firefly's movements, which makes every movement of firefly better.

V. EXPERIMENTS SETTINGS AND RESULTS

In this section, we first describe moving peaks benchmark [22] on which the proposed algorithms is evaluated. Then, experimental settings are described. Finally, experimental results of the proposed algorithm are presented and compared with alternative approaches from the literature. Moving peaks benchmark *"Fig. 3"* [22] is widely used in the literature to evaluate the performance of optimization algorithms in dynamic environments [23]. In this benchmark, there are some peaks in a multi-dimensional space, where the height, width, and position of each peak alter when the environment changes. Unless stated otherwise, the parameters of the moving peaks benchmark are set to the values presented in Table II. In order to measure the efficiency of the algorithms, offline error that is the average fitness of the best position found by the swarm at every point in time is used [24]. Offline error of proposed algorithm compared with other algorithm and it shown in Table I.

TABLE I. OFFLINE ERROR ±STANDARD ERROR

frequency of change		Cellular PSO	mQSO10	FMSO	Proposed firefly
5000	1	2.55±0.12	3.82±0.35	3.44±0.11	**0.9±0.1**
	5	1.68±0.11	1.90±0.08	2.94±0.07	**1.16±0.06**
	10	1.78±0.05	1.91±0.08	3.11±0.06	**1.38±0.09**
1000	1	6.77±0.38	18.6±1.6	14.42±0.4	**5.77±0.2**
	5	5.30±0.32	6.56±0.38	10.59±0.2	**4.07±0.1**
	10	5.15±0.13	5.71±0.22	10.40±0.1	**4.4±0.15**
500	1	13.4±0.74	33.67±3.4	27.58±0.9	**10.5±0.21**
	5	9.63±0.49	11.91±0.7	19.45±0.4	**7.65±0.2**
	10	9.42±0.21	9.62±0.34	18.26±0.3	**7.01±0.18**

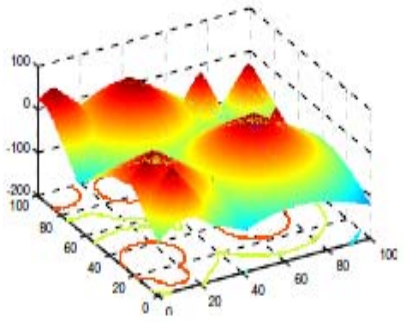

Figure 3-Moving peaks benchmark

TABLE II. DEFAULT SETTING OF MOVING PEAKS BENCHMARK

Value	Parameter
[0,10]	Number of peaks
500,1000,5000	frequency of change f
7	height severity
1	width severity
cone	peak shape
1	shift length s
5	number of dimensions D
[30,70]	cone height range H
[1, 12]	cone width range W
50	cone standard height I

In the algorithm, issue dimension is assumed 5, population size is 100 and number of iterations is 30. Reward Coefficients for automatons are 0.1 and penalty coefficients are 0.2. Three values of α in LA for gamma can be adjusted by initial number 0.5, Three values of α in LA for β_0 can be adjusted by initial number 0.1 and Three values of α in LA for alpha can be adjusted by initial number 0.1, 0 and -0.1. In equation 5, value of x is considered 9.0, y=1 and n=5. In any iteration for first and second LA, the second α(II) is multiplied with 1.003, β_0 may not be more than 1 and gamma may not be more than 4.

The proposed algorithm is compared with mQSO[25] and FMSO [27], and cellular PSO [28, 29]. For mQSO we adapted a configuration 10(5+5q) which creates 10 swarms with 5 neutral (standard) particles and 5 quantum particles with r_{cloud}=0.5 and r_{excl}= r_{conv}=31.5, as suggested in [25, 26]. For FMSO, there are at most 10 child swarms each has a radius of 25.0. The size of the parent and the child swarms are set to 100 and 10 particles, respectively [27]. For cellular PSO, a 5-Dimensional cellular automaton with 105 cells and Moore neighborhood with radius of two cells is embedded into the search space. The maximum velocity of particles is set to the neighborhood radius of the cellular automaton and the radius for the random local search (r) is set to 0.5 for all experiments. The cell capacity θ is set to 10 particles for every cell [28, 29].

VI. CONCLUSIONS

In the algorithm of optimizations initial parameters, from beginning to end, are constant and best values can be achieved by going through try and error. In order to deal with varying optimization problems, which can be applied to the real world problems, a combination of learning automata and firefly algorithm this proposed. This new approach is designed to enhance the performance of firefly algorithm in dynamic environment. This Algorithm evaluated on a variety of instances of the multimodal dynamic moving peaks benchmark. Results are also compared with standard firefly algorithm, standard PSO Algorithm and other PSO. They show that the new algorithm optimizer significantly outperforms previous approaches. Among the future works of the authors, learning automata can be used in order to do further research on adjusting other parameters such as r_{excl}, r_{cloud}.

REFERENCES

[1] X-S. Yang, "Firefly Algorithm, Stochastic Test Functions and Design optimization," International Journal of Bio-Inspired Computation, Vol. 2, No. 2, pp. 78-84(7), 2010.

[2] R.I. Lung, and D, Dumitrescu, "Evolutionary swarm cooperative optimization in dynamic, environments," Natural Computing 9, pp.83-94, 2020.

[3] D. Pelta, C. Cruz and J.R. González, "A study on diversity and cooperation in a multiagent," strategy for dynamic optimization problems, International Journal of Intelligent Systems 24, pp.844–861, 2009.

[4] H. Wang, D. Wang and S.Yang, "A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems," Soft Computing-A Fusion of Foundations, Methodologies and Applications 13, pp.763–780, 2009.

[5] X-S.Yang and X. Yao, "Population-Based Incremental Learning With Associative Memory for Dynamic Environments," IEEE Trans. Evol. Computat. 12, pp.542-561, 2008.

[6] R. Brits, A. Engelbrecht and F. van den Bergh, "A niching particle swarm optimizer," in Proc. Asia-Pacific Conf. Simulated Evol. Learning, pp. 692–696, 2002.

[7] X. Li, "Adaptively choosing neighborhood bests using species in a particle swarm optimizer for multimodal function optimization," in Proc. Genetic Evol. Comput. Conf, vol. 3102, K. Deb et al., Eds., pp.105–116, 2004.

[8] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," IEEE Trans. Evol. Comput., vol. 8, no. 3, pp. 225–239, 2004.

[9] Y. Shi and R. A. Krohling, "Co-evolutionary particle swarm optimization to solve min-max problems," in Proc. Congr. Evol. Comput, pp. 1682–1687, 2002.

[10] J. Kennedy, "Stereotyping: Improving particle swarm performance with cluster analysis," in Proc. Congr. Evol. Comput., pp. 1507–1512, , 2000.

[11] T. Blackwell, "Swarm music: improvised music with multi-swarms," in Proc. AISB Symp. Artif. Intell. Creativity Arts Sci., Aberystwyth, U.K., Society for the Study of Artificial Intelligence and the Simulation of Behavior, Univ. Wales, pp. 41–49, 2003.

[12] T. Blackwell and J. Branke, "Multiswarm, Exclusion, and Anti-Convergence in Dynamic Environment," in IEEE Transaction on Evolutionary Computation, Vol. 10, No. 4, pp. 459-472, 2006.

[13] H. Beigy and M. R. Meybodi, "An Adaptive Call Admission Algorithm for Cellular Networks," Journal of Computer and Electrical Engineering, vol. 31, no. 2, pp. 132-151, 2005.

[14] H. Beigy and M. R. Meybodi, "Call Admission Control in Cellular Mobile Networks: A Learning Automata Approach," in Eurasia-Ict 2002: Information and Communication Technology, Lecture Notes in Computer Science, vol. 2510, pp. 450-457, 2002.

[15] B. J. Oommen and T. D. Roberts, "Continuous Learning Automata Solutions to the Capacity Assignment Problem," IEEE Transactions on Computers, vol. 49, no. 6, pp. 608-620, 2000.

[16] M. R. Meybodi and H. Beigy, "A Note on Learning Automata Based Schemes for Adaptation of Bp Parameters," Journal of Neurocomputing, vol. 48, no. 4, pp. 957-974, 2002.

[17] H. Beigy and M. R. Meybodi, "A Learning Automata Based Algorithm for Determination of the Number of Hidden Units for Three Layers Neural Networks," International Journal of Systems Science, to appear, vol. 2510, pp. 450-457, 2002.

[18] Ying Liu, Aixin Sun and Han Tong Loh, "Advances of Computational Intelligence in Industrial Systems," books.google.com, PP.27, 2008.

[19] X-S. Yang, "Firefly Algorithm, Levy Flights and Global Optimization," Research and Development in Intelligent Systems XXVI (Eds M. Bramer, R. Ellis, M. Petridis), Springer London, pp. 209-218, 2010.

[20] K. S. Narendra and M. A. L. Thathachar, " Learning Automata: an Introduction: Prentice-Hall Inc," 1989.

[21] B. Masoumi and M. R. Meybodi, "Learning Automata based Multi-agent System Algorithms for Finding Optimal Policies in Markov Games," Asian Journal of Control, 2010.

[22] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problem," In: 1999 Congress on Evolutionary Computation, Washington D.C., USA, vol. 3, pp.1875–1882, 1999.

[23] I. Moser, "All Currently Known Publications On Approaches Which Solve the Moving Peaks Problem. Swinburne University of Technology," Melbourne, Australia, 2007.

[24] J. Branke, H. Schmeck, "Designing evolutionary algorithms for dynamic optimization problems. Advances in evolutionary computing: theory and applications," pp.239–262, 2003.

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

[26] T. Blackwell, J. Branke, X. Li, "Particle Swarms for Dynamic Optimization Problems. Swarm Intelligence," pp.193–217, 2008.

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

[28] A.B. Hashemi, M.R. Meybodi, "Cellular PSO: A PSO for Dynamic Environments. Advances in Computation and Intelligence," 422–433, 2009.

[29] A.B. Hashemi, M.R. Meybodi, " A multi-role cellular PSO for dynamic environments," In: 14th International CSI Computer Conference, Tehran, Iran, pp. 412–417, 2009.

[30] S. Lukasik, S. Zak, "Firefly Algorithm for Continuous Constrained Optimization Tasks," First International Conference, ICCCI 2009, Wroclaw, Poland , vol 5796, pp. 97-106, 2009