Tracking Extrema in Dynamic Environments Using a Learning Automata-based Immune Algorithm

Alireza Rezvanian[1] and Mohammad Reza Meybodi[2]
[1] Department of Computer Engineering, Islamic Azad University, Hamedan branch, Iran
[2] Department of Computer & IT Engineering, Amirkabir University of Technology, Tehran, Iran

rezvan@iauh.ac.ir, mmeybodi@aut.ac.ir

Abstract. In recent years, bio-inspired algorithms have been used by researchers for solving various optimization problems increasingly. Many real world problems are mostly time varying optimization problems, which require special mechanisms for detect changes in environment and then response to them. The paper has been proposed combination of learning automata and artificial immune algorithm, in order to improve the performance of immune system algorithm in dynamic environments. In the proposed algorithm, the immune cells have equipped with a learning automaton. So they can increase diversity in response the dynamic environments. Learning automata based immune algorithm for dynamic environment has been tested in the moving parabola as a popular standard dynamic environment, and compared by several famous algorithms in dynamic environments.

Keywords: Artificial Immune Algorithm, Learning Automata, Dynamic Environments, Time varying problems.

1 Introduction

Most of the real world problems are known as optimization problems. In these problems, the extrema change during the time whole the landscape [1]. 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 for solving different optimization problems such as dynamic optimization problems [2]. 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 for dealing with time varying optimization have been proposed based on population based algorithms in [3-5].

Artificial Immune Systems (AIS) are bio-inspired algorithms that take their inspiration from complex defensive mechanism of human immune system for protecting against pathogens [6, 26]. The immune algorithms have been used for many different applications such as optimization problems [7-13]. More recently, also in [4] and [14], AIS has been implemented for optimization in dynamic environments. Moreover, the development of these algorithms has not been widespread and they are not much developed in comparison with other population-based methods such as

Particle Swarm Optimization (PSO) [1, 15]. Learning Automaton (LA) is a general purpose stochastic optimization tool with finite states, which has been developed as a model for learning systems. The LA tries to determine, iteratively, the optimal action to apply to the environment from a finite number of actions that are available to it. The environment returns a reinforcement signal that shows the relative quality of an action of the LA, and then LA adjusts itself by means of a learning algorithm [7, 16]. Previously, LA have been used successfully in many applications such as evolutionary algorithms, in order to enhance the learning and adaptation abilities different parameters in the genetic algorithm (GA) [17], in PSO [8], Ant Colony Optimization (ACO), and recently in AIS [19] . So this paper presents a new method to empower the AIS to increasing diversity in react changes in dynamic environment by use of LA as a solution to improving AIS in dynamic environments. In the rest of the paper is organized as follow: next section briefly introduces Immune Algorithms (IA). LA will be presented in the third section. The forth section consider the proposed algorithms, finally, in the last section the results of simulation for the dynamic environments of moving parabola are presented.

2 Immune Algorithms

Immune algorithms are bio-inspired algorithms which have been inspired by human immune system [6, 26]. Human immune system is divided into innate immune and adaptive immune, the algorithms modeled based on the latter. In addition, According to the different theories for natural immune system, the inspired algorithms are categorized into several groups: negative selection, clonal selection, bone marrow, immune network, and danger theory, which are utilized with wide variety of application such as optimization in static environments [7]. Theoretically, the mechanism human immune system can handle well the changes in environment and react to these changes, however, the AIS algorithms unable to trace the changes in dynamic environments, due to its lack some mechanism are needed in AIS algorithm to detect and response the changes [14]. In [20] the immune network algorithm has undergone some changes to be used in dynamic environments. In [4] after the comparison of different mutations for immune network and clonal selection, performance of these algorithms has been compared with them.

3 Learning Automata

Learning automaton (LA) has finite set of actions and at each stage chooses one of them. The choice of an action depends on the state of LA represented by an action probability vector. For the action chosen by the LA, the environment gives a reinforcement signal with unknown probability distribution. Then based on this signal, the LA updates its action probability vector using a learning algorithm. A class of LA called *variable structure learning automata* are represented by a triple $\langle \beta, \alpha, T \rangle$ where $\beta = \{0, 1\}$ is a set of inputs, $\alpha = \{\alpha_1, ..., \alpha_r\}$ is a set of actions, and *T* is the

learning algorithm [27]. In *linear reward-inaction learning algorithm* (L_{R-I}), the action probability vector is updated using equation (1):

$$p_j(k+1) = \begin{cases} p_j(k) + b \times [1 - p_j(n)] & i = j \\ p_j(k) - b \times p_j(k) & i \neq j \end{cases} \quad \textbf{(1)}$$

When the environment rewards and the action probability vector remains unchanged when the environment penalizes the action. Parameters b $\in$ (0, 1) and r represent learning parameter and the number of actions for LA, respectively and α_i is the action chosen at stage k as a sample realization from probability distribution p(k) [28]. LA have been used successfully in many applications such as control of broadcast networks, intrusion detection in sensor networks, database systems, and solving shortest path problem in stochastic networks, to mention a few [16]. The interaction between LA and environment is shown in Figure 1.

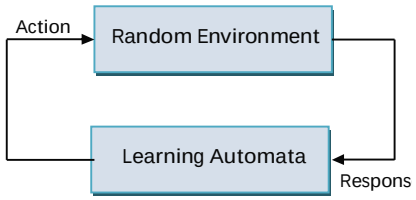


Fig. 1. The interaction between the learning automata and its environment

4 Proposed Algorithm: Learning Automata based Immune Algorithm

In IA, mutation, as the only and most important operator, should be act effectively, and it is also termed hyper mutation. The immune cells in immune algorithm suffer the mutation according the probability of mutation rate, which calculated by a Gaussian distribution following equation (2):

$$c' = c + \alpha \times N(0,1), \quad \alpha = \frac{1}{\beta} \times exp(-f^*) \quad \textbf{(2)}$$

Where in equation (1), c' represents mutated cells, and N(0,1) is a Gaussian distribution with a mean of 0 and standard deviation of 1. Here beta is taken as the decay parameter for control the inverse exponential function and f* is the fitness of an individual normalized [20].

Hyper mutation in this algorithm is considered as probabilistic and with an affinity between antibody and antigen. Therefore, cells with highest affinity suffer the lowest mutation rate, whereas the lowest affinity cells has high mutation rate. Probability of mutation rate is obtained from equation (3) [21].

$$P_m = \begin{cases} \alpha\left(0.5 - f_d^2\right), & 0 \le f_d \le 0.5, \quad 0 < \alpha \le 1.0 \\ \alpha\left(1 - f_d^2\right)^2, & 0.5 \le f_d \le 1.0, \quad 0 < \alpha \le 1.0 \end{cases} \tag{3}$$

Where p_m is hypermutation rate with a value less than 0.5, α is the scale coefficient with a value less than 0.1 and f_d is the value of normalized fitness. Scale coefficient cannot be assigned a precise value; therefore the LA is used for adaptive setting of the parameter of α scale coefficient. Three actions, namely "*increasing* α", "*decreasing* α" and "*fixing* α" are considered for LA. In each step, LA selects one action according to the probabilistic vector and thus the value of the parameter of scale coefficient becomes modified. Consequently, hypermutation rate mutates antibodies by means of the new value of the scale coefficient. Afterwards, based on evaluating the performance, the probabilistic vector of action becomes up to date. At the beginning, the probability of selecting each action for probabilistic vector is initialized equal. Then based on the selected action and the feedback from the actions will update in the each step. The general structure of LA has been shown in figure 2.

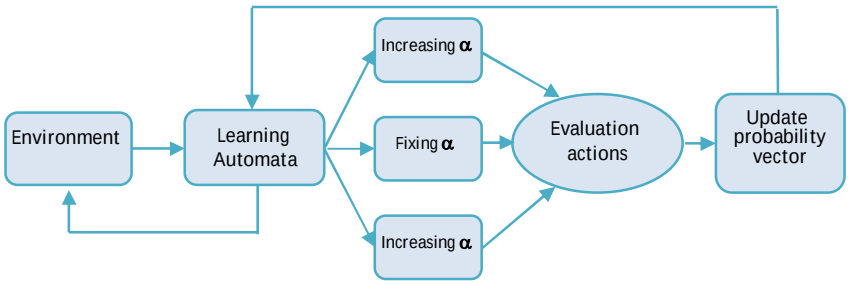


Fig. 2. The structure of the proposed learning automata

At the end of each step, based on the density between individuals, some parts of population are eliminated. The population density of immune cells is relative to the distance between cells. In every step the distance between two antibodies is calculated by Euclidian distance following in equation (4) [19].

$$D = \left[\sum_{i=1}^{n} \left| A(x_i) - B(x_i) \right|^2 \right]^{1/2} \tag{4}$$

Suppose a threshold T, if D < T, two immune cells are considered much close to each other, so the one with lower fitness is removed.

In dynamic environments, after detecting environmental changes by memory cells, the reaction of the algorithm against these changes is considered reset the parameters. Certainly, the initial value of the parameters is not appropriate and the LA can find a proper value of parameters in a stochastic environment.

Now, regarding the above-mentioned points, the steps of the proposed algorithm for dynamic environments can be formed as following pseudo-code in figure 3:

```
1. Initialize immune cells and the probabilistic vector for LA.
2. Repeat the following steps for every immune cell until reach stopping
criteria.
  2.1. Evaluate the fitness function of immune cells.
  2.2. If the change is detected in the environment:
    2.2.1. Re-initialize the immune cells and the probabilistic vector
     of LA.
    2.2.2. Re-evaluate the fitness function of immune cells.
  2.3. Select the best immune cell.
  2.4. Select one action according to the probabilistic vector of LA.
  2.5. Clone and Mutate immune cells according to the probability
   mutation rate obtained from the selected action of the LA
  2.6. Evaluate the performance of the selected action and update the
   probabilistic vector of the LA.
  2.7. Retain the best antibodies as memory.
  2.8. Calculate the distance between cells.
  2.9. Remove a part of weak immune cells.
  2.10 Replace weak immune cells with new immune cells.
3. Repeat the steps.
```

Fig. 3. The pseudo-code of proposed algorithm

If there is no change in the environment, the method for evaluating the selected action for the automata is followed: the average performance of immune cells in the current state is compared with the average performance of immune cells in the previous state; If the state is improved, the selected action is evaluated as positive, and if the changes are insignificant, it is not logical to change the probabilistic vector of automata, and otherwise it is evaluated as negative.

Among the important advantages of this method is adaptation of the parameter of scale coefficient according to the environmental changes to increase diversity as well as its high ability in stable environments to escape from local optima or proper convergence. In fact, increase in α leads to expansion of mutation radius and a global search, but decrease in α leads to reduction of mutation radius and a local search in the search space.

2 Experimental Results

In order to evaluate the efficiency of the results of the algorithm, Offline Error (oe) is the famous criterion known as one of the most popular criteria in different papers for evaluating the average the deviation of the best value since the last change from the optimum is computed by equation (5) [4]:

$$Offline\ Error = \frac{1}{N_c}\sum_{j=1}^{N_c}\left(\frac{1}{N_e(j)}\sum_{i=1}^{N_e(j)}\left(f_j^* - f_{ji}^*\right)\right) \quad (5)$$

Where N_c is the total number of fitness landscape changes within a single experiment, $N_e(j)$ is the number of solutions evaluations performed for the j^{th} state of the environment, f^*_j, is the value of optimal solution for the j^{th} landscape, and f^{*ji} is the current best fitness value found for the j^{th} landscape [4].

Since most real world problems are dynamic spontaneously, so the environments change within the time, for evaluating the proposed algorithm in uncertainty

environments termed Angeline [22] known as the moving parabolas mentioned in [23] and [24] was used. In moving parabolas three types of movement, namely linear, circular and Gaussian, are present, and changes are controlled by two parameters of step size (τ) and change frequently (f). The movement in Angeline method is calculated for linear, circular and Gaussian respectively by the equations as follows:

$$\Delta k = \Delta k + \tau \tag{6}$$

$$\Delta k = \begin{cases} \Delta k + \tau \sin\left(\frac{2\pi t}{25}\right) & k \text{ is even} \\ \Delta k + \tau \cos\left(\frac{2\pi t}{25}\right) & k \text{ is odd} \end{cases} \tag{7}$$

$$\Delta k = \Delta k + N(0,1) \tag{8}$$

Δk is added to each variable as an update parameter. In order to evaluate the proposed method, the four benchmark functions Rastrigin, Griewank, Rosenbrock, and Sphere was used according to the conditions mentioned in table 1 [7]. Initial values and parameters of moving parabolas, according to [20], were set as τ=0.1, f=1, and d=30, with a total of 1000 iterations and average of 10 independent experiments.

Table 1. The benchmark functions and the initialization range

name	Range	Dimension (N)	Function
Rastrigin	$[-5.12, 5.12]^n$	30	$f_2(x) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$
Griewank	$[-600, 600]^n$	30	$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
Rosenbrock	$[-100, 100]^n$	30	$f_4(x) = \sum_{i=1}^{n-1} \left(100(x_{i+1} - x_i)^2 + (x_i - 1)^2\right)$
Sphere	$[-1.28, 1.28]^n$	30	$f_1(x) = \sum_{i=1}^{n} x_i^2$

Among these functions, Sphere and Rosenbrock are well known as single-modal, Rastrigin and Griewank are known as multi-modal [8].

In order to make a comparison based on OE and standard deviation of OE, in additional to the proposed method termed Lopt-aiNet, use was also made of immune network algorithm method named opt-aiNet [25], dynamic immune network algorithm called Dopt-aiNet [20] and the multi-population immune network algorithm named Mopt-aiNet [14].

The results of these experiments, according to the mentioned conditions, are presented in table 2 to 5 respectively for Sphere, Rastrigin, Griewank, and Rosenbrock.

Table 2. The Comparison of the OE of the proposed algorithm with different algorithms in dynamic environment for Sphere function.

Algorithm	OE±STD		
	Linear	**Circular**	**Gaussian**
Opt-aiNet	1.21±1.13	1.35±1.79	1.26±1.37
Dopt-aiNet	0.02±0.22	0.32±0.18	0.02±0.02
Mopt-aiNet	0.03±0.08	0.14±0.09	0.03±0.03
Lopt-aiNet	0.02±0.05	0.15±0.07	0.01±0.02

Table 3. Comparison of the OE of the proposed algorithm with different algorithms in dynamic environment for Rastrigin function.

Algorithm	OE±STD		
	Linear	**Circular**	**Gaussian**
Opt-aiNet	3.13±2.24	5.12±2.28	3.91±1.74
Dopt-aiNet	0.50±0.17	0.57±0.24	0.22±0.17
Mopt-aiNet	0.11±0.08	0.14±0.39	0.17±0.13
Lopt-aiNet	0.12±0.07	0.15±0.33	0.19±0.09

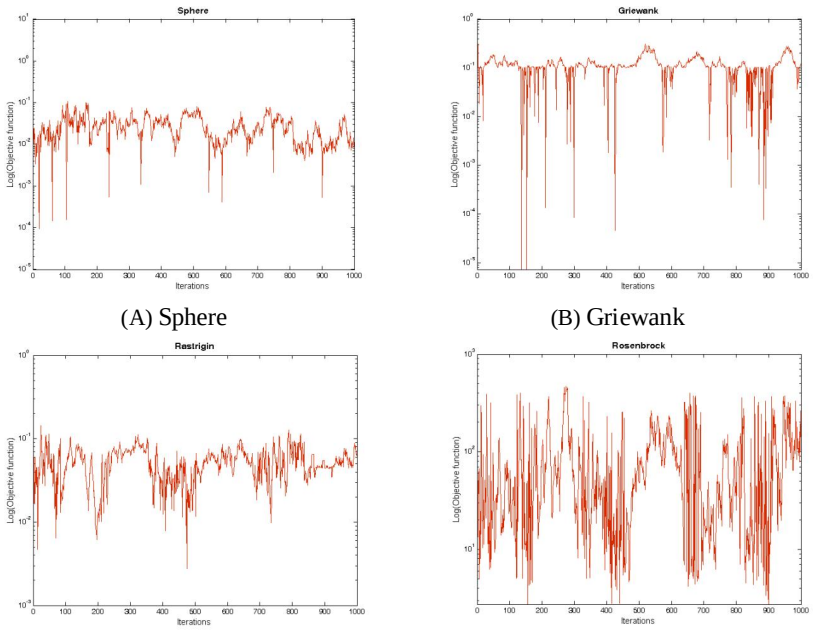
Table 4. Comparison of the OE of the proposed algorithm with different algorithms in dynamic environment for Griewank function.

Algorithm	OE±STD		
	Linear	**Circular**	**Gaussian**
Opt-aiNet	1.93±2.01	1.77±1.93	0.96±1.78
Dopt-aiNet	0.13±1.76	0.33±0.17	7.57±5.79
Mopt-aiNet	0.03±0.12	0.29±0.12	0.02±0.13
Lopt-aiNet	0.03±0.02	0.31±0.07	0.02±0.02

Table 5. Comparison of the OE of the proposed algorithm with different algorithms in dynamic environment for Rosenbrock function.

Algorithm	OE±STD		
	Linear	**Circular**	**Gaussian**
Opt-aiNet	1.53±1.91	4.81±1.25	1.38±1.26
Dopt-aiNet	0.03±0.16	0.38±0.58	0.03±0.16
Mopt-aiNet	0.10±0.82	0.27±0.12	0.41±2.13
Lopt-aiNet	0.03±0.12	0.32±0.12	0.01±0.02

In figure 4, the behavior of the proposed algorithm for different function is shown in a logarithmic form as shown, the algorithm can very well follow the optima and some oscillations are observed, but Rosenbrock function due to its many plateaus has more oscillations than do other functions.



(A) Sphere (B) Griewank

(C) Rastrigin (D) Rosenbrock

Fig. 4. Logarithmic behavior of the proposed algorithm on dynamic environments (t= 0.1, f= 1.0 and circular displacement).

5 Conclusion

This paper proposes a combination of the LA and AIS in order to improve the standard immune algorithm in dynamic environments. Since after detecting an environmental change there is no guarantee for the parameters of execution of the standard algorithms, the values of the parameters of the probabilistic of hypermutation rate are set and thus the LA approaches the proper values by feedback from the environment. The simulation results of the proposed algorithm on the moving parabolas as linear, circular and Gaussian displacement show that this algorithm is relatively more improvement than other methods based on immune algorithm. Among the future works of the authors, assessing other methods of hypermutation in the proposed algorithm is to be mentioned.

References

1. Lung, R.I. & Dumitrescu, D. Evolutionary swarm cooperative optimization in dynamic environments. Natural Computing 9, 83–94 (2010).

2. Pelta, D., Cruz, C. & González, J.R. A study on diversity and cooperation in a multiagent strategy for dynamic optimization problems. International Journal of Intelligent Systems 24, 844–861 (2009).
3. Wang, H., Wang, D. & Yang, S. A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems. Soft Computing-A Fusion of Foundations, Methodologies and Applications 13, 763–780 (2009).
4. Trojanowski, K. & Wierzchon, S.T. Immune-based algorithms for dynamic optimization. Information Sciences 179, 1495–1515 (2009).
5. Shengxiang Yang & Xin Yao Population-Based Incremental Learning With Associative Memory for Dynamic Environments. IEEE Trans. Evol. Computat. 12, 542-561 (2008).
6. de Castro, L. & Von Zuben, F. Learning and optimization using the clonal selection principle. IEEE Trans. Evol. Computat. 6, 239-251 (2002).
7. Rezvanian, A. and Meybodi, M.R. LACAIS: Learning Automata based Cooperative Artificial Immune System for Function Optimization. 3rd International Conference on Contemporary Computing (IC3 2010), Part I, 64-75 (2010).
8. Rezvanian, A. and Meybodi, M. R., Improving Artificial Immune System using Fuzzy Logic. Proceedings of the 10th Conference on Fuzzy Systems (IFS 2010), 173-177 (2010).
9. Khaled, A., Ab d ul-Kader, H.M. & Ismail, N.A. Artificial Immune Clonal Selection Algorithms: A Comparative Study of CLONALG, opt-IA, and BCA with Numerical Optimization Problems. IJCSNS 10, 24 (2010).
10. Aragón, V.S., Esquivel, S.C. & Coello, C.A. Artificial Immune System for Solving Global Optimization Problems. Inteligencia artificial 14, 46 (2010).
11. Xu, Q., Wang, L. & Si, J. Predication based immune network for multimodal function optimization. Engineering Applications of Artificial Intelligence, 23, 495-504 (2010).
12. Gao, J. & Wang, J. WBMOAIS: A novel artificial immune system for multiobjective optimization. Computers & Operations Research 37, 50–61 (2010).
13. Verbeeck, K. & Nowe, A. Colonies of learning automata. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 32, 772–780 (2002).
14. Xuhua, S. & Feng, Q. An Optimization Algorithm Based on Multi-population Artificial Immune Network. 5th International Conference on Natural Computation 379 - 383 (2009).
15. Hu, C., Wang, B. & Wang, Y. Multi-swarm particle swarm optimiser with Cauchy mutation for dynamic optimisation problems. International Journal of Innovative Computing and Applications 2, 123–132 (2009).
16. Beigy, H. & Meybodi, M.R. Cellular learning automata with multiple learning automata in each cell and its applications. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 40, 54–65 (2010).
17. Abtahi, F., Meybodi, M.R., Ebadzadeh, M.M. & Maani, R. Learning automata-based co-evolutionary genetic algorithms for function optimization. 6th International Symposium on Intelligent Systems and Informatics, 2008. SISY 2008, 1–5 (2008).
18. Hamidi, M. & Meybodi, M.R. New Learning Automata based Particle Swarm Optimization Algorithms. Proceedings of the 2nd Iranian Data Mining Conference, Amirkabir University of Technology, Tehran, Iran, (2008).
19. Rezvanian, A. and Meybodi, M. R., A New Method for Function Optimization using Artificial Immune System and Learning Automata. Proceedings of the Third Joint Congress on Fuzzy and Intelligent Systems, University of Yazd (2009).
20. de Franca, F.O., Von Zuben, F.J. & de Castro, L.N. An artificial immune network for multimodal function optimization on dynamic environments. Proceedings of the 2005 conference on Genetic and evolutionary computation 296 (2005).
21. Yongshou, D., Yuanyuan, L., Lei, W., Junling, W. & Deling, Z. Adaptive immune-genetic algorithm for global optimization to multivariable function. Journal of Systems Engineering and Electronics 18, 655–660 (2007).

22.Angeline, P. Tracking extrema in dynamic environments. Evolutionary Programming VI 335–345 (1997).
23.Woldesenbet, Y.G. & Yen, G.G. Dynamic evolutionary algorithm with variable relocation. IEEE Transactions on Evolutionary Computation 13, 500–513 (2009).
24.Dempsey, I., O'Neill, M. & Brabazon, A. Foundations in Grammatical Evolution for Dynamic Environments. Springer Verlag (2009).
25.Walker, J.H. & Garrett, S.M. Dynamic Function Optimisation: Comparing the Performance of Clonal Selection and Evolution Strategies. Artificial immune systems: second international conference, ICARIS 2003, Edinburgh, UK, proceedings 273 (2003).
26. Timmis, J., Hone, A., Stibor, T. and Clark, E. Theoretical advances in artificial immune systems. Theoretical Computer Science, 403, 11–32 (2008).
27. Narendra, K.S. and Thathachar, M.A.L. Learning Automata: An Introduction.: Printice-Hall, New York (1989).
28. Beigy, H. and Meybodi, M.R. Asynchronous cellular learning automata. Automatica, 44, 1350–1357 (2008).