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 increasingly 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 detecting changes in environment and then responding to them. The present paper has been proposed to combination the 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 are 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 real world problems are known as optimization problems. In these problems, the extrema change along the landscape during time [1]. Due to lack of traditional optimization algorithms to trace the optima they failed in the non-stationary environments. Therefore, bio-inspired algorithms are more frequently used for solving different optimization problems such as dynamic optimization problems [2]. In dynamic optimizations, not only do the algorithms have to detect the changes but also must respond to the changes in the landscape. Several suggestions which have been proposed for dealing with time varying optimization based on population based algorithms in [3-5].

Artificial Immune Systems (AIS) are bio-inspired algorithms that take their inspiration from the 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. It has been developed as a model for learning systems. 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, LAs were successfully used in many applications such as evolutionary algorithms, in order to enhance the learning and adaptation abilities of different parameters in genetic algorithm (GA) [17], PSO [8], Ant Colony Optimization (ACO), and recently in AIS [19]. So this paper presents a new method to improve the AIS increasing diversity which occurs when changes in dynamic environment happen. This is done by the use of LA as a solution for improving AIS in dynamic environments. The rest of the paper is organized as follows: the next section briefly introduces Immune Algorithms (IA). LA will be presented in the third section. The forth section considers the proposed algorithms and 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 have been modeled based on the latter. In addition, according to 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 a wide variety of application such as optimization in static environments [7]. Theoretically, the mechanism of the human immune system can handle well the changes in the environment and can react to these changes; however, the AIS algorithms are unable to trace the changes in dynamic environments. As a result, some mechanisms are needed in AIS algorithm to detect and respond to 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 of immune network and clonal selection, the performance of these algorithms has been compared with each other.

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):

$$p_j(k+1)=\begin{cases} p_j(k)+b\times\lfloor 1-p_j(n)\rfloor & i=j \\ p_j(k)-b\times p_j(k) & i\neq j \end{cases} \tag{1}$$

When the environment rewards and the action probability vector remains unchanged, 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 successfully used 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.

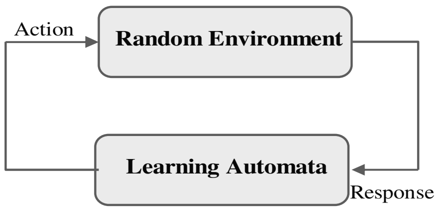


Fig. 1. The interaction between the learning automata and its environment

4 The Proposed Algorithm: Learning Automata Based Immune Algorithm

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

$$\begin{aligned} c' &= c+\alpha\times N(0,1), \\ \alpha &= \frac{1}{\beta}\times exp(-f^{*}) \end{aligned} \tag{2}$$

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 β is taken as the decay parameter to control the inverse exponential function and f^* is the fitness of an individual normalized [20].

Hypermutation 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. The 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, according the selected action and the feedback from the actions it will be update in each step. The general structure of LA has been illustrated in figure 2.

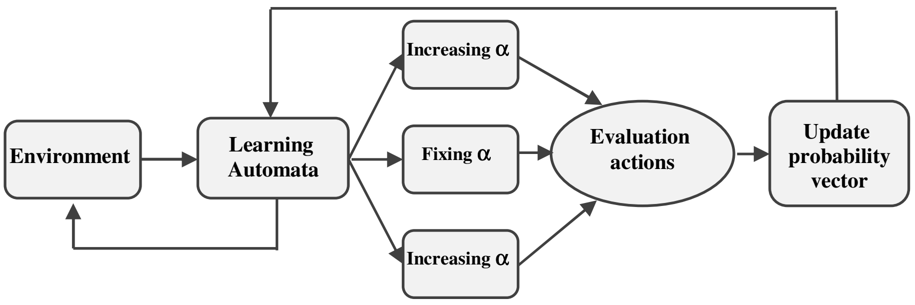


Fig. 2. The structure of the proposed learning automata

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 the two antibodies is calculated by Euclidian distance as follows in equation (4) [19].

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

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

In dynamic environments, after detecting environmental changes by memory cells, it is considered that the reaction of the algorithm against these changes is to 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 reaching
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 as follows: 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 the adaptation of the scale coefficient parameter according to the environmental changes in order to increase its diversity and 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.

5 Experimental Results

In order to evaluate the efficiency of the results of the algorithm, Offline Error (oe) was used which is the famous criterion in different papers for evaluating the average deviation of the best value since the last change from the optimum. It is computed by equation (5) [4]:

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

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* method [22] was used. It is also known as the *moving parabolas* mentioned in [23] and [24]. 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 frequency (*f*). The movement in Angeline method is calculated for linear, circular and Gaussian respectively by the following equations:

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

$$\Delta k' = \begin{cases} \Delta k + \tau \mathit{sin}\left(\dfrac{2\pi t}{25}\right) & k \text{ is even} \\ \Delta k + \tau \mathit{cos}\left(\dfrac{2\pi t}{25}\right) & k \text{ is odd} \end{cases} \tag{7}$$

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

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*=1.0, and *d*=30, with a total of 1000 iterations and an 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 addition to the proposed method which is termed *Lopt-aiNet*, use was also made of an 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 that of 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 that of 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

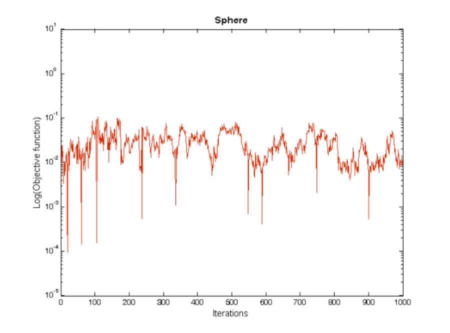
Table 4. Comparison of the OE of the proposed algorithm with that of different algorithms in dynamic environment for *Griewank* function

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

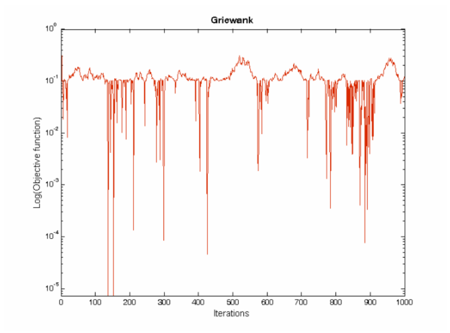
Table 5. Comparison of the OE of the proposed algorithm with that of different algorithms in dynamic environment for *Rosenbrock* function

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

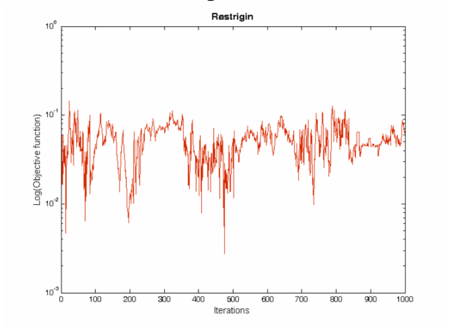
In figure 4, the behavior of the proposed algorithm for different functions is shown in a logarithmic form as shown; the algorithm can very well follow the optima and some oscillations are observed, but Rosenbrock function has more oscillations than do other functions due to its many plateaus.



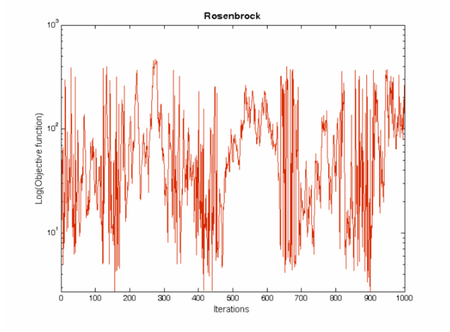
(A) Sphere



(B) Griewank



(C) Rastrigin



(D) Rosenbrock

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

6 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 parameter values of execution of the standard algorithms, the parameter values of the probabilistic of hypermutation rate are set and thus the LA approaches the proper values by feedback from the environment. The main advantage of the proposed method is that it does not require parameter tuning. The simulation results of the proposed algorithm on the moving parabolas as linear, circular and Gaussian displacement show that this algorithm is relatively more improved 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.

Acknowledgments. The authors thank Mr. Arash Moradi and Mr. Masoud Safaee for some improvements.

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., Gonzalez, 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. Yang, S., Yao, X.: Population-based Incremental Learning with Associative Memory for Dynamic Environments. IEEE Transactions on Evolutionary Computation 12, 542–561 (2008)
6. De Castro, L.N., Von Zuben, F.J.: Learning and Optimization using the Clonal Selection Principle. IEEE Transactions on Evolutionary Computation 6, 239–251 (2002)
7. Rezvanian, A., Meybodi, M.R.: LACAIS: Learning Automata based Cooperative Artificial Immune System for Function Optimization. In: Ranka, S., et al. (eds.) 3rd International Conference on Contemporary Computing (IC3 2010), Noida, India. Contemporary Computing, CCIS, vol. 94, pp. 64–75. Springer, Heidelberg (2010)
8. Rezvanian, A., Meybodi, M.R.: Improving Artificial Immune System using Fuzzy Logic. In: 10th Iranian Conference on Fuzzy Systems (IFS 2010), Tehran, Iran, pp. 173–177. IFSS, Tehran (2010)
9. Khaled, A., Abdul-Kader, H.M., Ismail, N.A.: Artificial Immune Clonal Selection Algorithms: A Comparative Study of CLONALG, opt-IA, and BCA with Numerical Optimization Problems. International Journal of Computer Science and Network Security 10, 24–30 (2010)
10. Aragon, V.S., Esquivel, S.C., Coello, C.A.: Artificial Immune System for Solving Global Optimization Problems. Inteligencia Artificial 46, 3–16 (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. In: 5th International Conference on Natural Computation (ICNC 2009), vol. 5, pp. 379–383. IEEE Press, New York (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. In: 6th International Symposium on Intelligent Systems and Informatics (SISY 2008), Subotica, Serbia, pp. 1–5. IEEE Press, New York (2008)

18. Hashemi, A.B., Meybodi, M.R.: Adaptive Parameter Selection Scheme for PSO: A Learning Automata Approach. In: 14th International CSI Computer Conference (CSICC 2009), Tehran, Iran, pp. 403–411. IEEE Press, New York (2009)
19. Rezvanian, A., Meybodi, M.R.: A New Method for Function Optimization using Artificial Immune System and Learning Automata. In: 3rd Joint Congress on Fuzzy and Intelligent Systems (IFS 2009), Yazd, Iran, pp. 1–7. IFSS, Tehran (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. In: Conference on Genetic and Evolutionary Computation (GECCO 2005), Washington, DC, USA, pp. 289–296. ACM, New York (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. In: Angeline, P.J., McDonnell, J.R., Reynolds, R.G., Eberhart, R. (eds.) EP 1997. LNCS, vol. 1213, pp. 335–345. Springer, Heidelberg (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, Heidelberg (2009)
25. Walker, J.H., Garrett, S.M.: Dynamic Function Optimisation: Comparing the Performance of Clonal Selection and Evolution Strategies. In: Timmis, J., Bentley, P.J., Hart, E. (eds.) ICARIS 2003. LNCS, vol. 2787, pp. 273–284. Springer, Heidelberg (2003)
26. Timmis, J., Hone, A., Stibor, T., Clark, E.: Theoretical Advances in Artificial Immune Systems. Theoretical Computer Science 403, 11–32 (2008)
27. Narendra, K.S., Thathachar, M.A.L.: Learning Automata: An Introduction. Prentice-Hall, New York (1989)
28. Beigy, H., Meybodi, M.R.: Asynchronous Cellular Learning Automata. Automatica 44, 1350–1357 (2008)