An Adaptive Mutation Operator for Artificial Immune Network Using Learning Automata in Dynamic Environments

Alireza Rezvanian
Department of Computer Engineering
Islamic Azad University, Hamedan branch
Hamedan, Iran
rezvan@iauh.ac.ir

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

Abstract—**Many real world problems are mostly time varying optimization problems, which require special mechanisms for detection changes in environment and then response to them. This paper proposes a hybrid optimization method based on learning automata and artificial immune network for dynamic environments, in which the learning automata are embedded in the immune cells to enhance its search capability via adaptive mutation, so they can increase diversity in response to the dynamic environments. The proposed algorithm is employed to deal with benchmark optimization problems under dynamic environments. Simulation results demonstrate the enhancements of our algorithm in tracking varying optima.**

Keywords-component; Artificial Immune Network, Learning Automata, Dynamic Environments, Dynamic Optimization problems

I. INTRODUCTION

In recent years, optimization problem has been a popular research subjects because many real world problems are known as optimization problems. The objective function changes over time in search space, cause to existing optima are disappeared [1]. Due to the lack of traditional optimization algorithms to trace the optima, the bio-inspired algorithms have been widely used for solving different optimization problems such as dynamic optimization problems [2, 7]. In dynamic optimizations, the algorithms not only have to detect the changes but also they must response to the changes in the search space [3-5]. There are several bio-inspired algorithms for dealing with time varying optimization such as Particle Swarm Optimization (PSO) [6] Ant Colony Optimization (ACO) [24] and Artificial Immune System (AIS) [4, 7-9]. The Artificial Immune network (AIN) is another method for solving dynamic optimization. AIN is a bio-inspired algorithm that takes their inspiration from complex defensive mechanism of human immune system for protecting against pathogens [10, 11]. In [4] and [9], AIN has been implemented for optimization in dynamic environments. However, the developments of these algorithms are not remarkable for dynamic environments. 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 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 [12]. 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) [13], in PSO [14], ACO [15], and recently in AIS [12]. So this paper presents a new method to empower the AIN 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 DF1 are presented.

II. IMMUNE ALGORITHMS

Immune algorithms are bio-inspired algorithms which have been inspired by human immune system [10,11]. 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 [12]. Theoretically, the human immune system mechanism can handle well the changes in the 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 [9]. In [8] the immune network algorithm has undergone some changes to be used in dynamic environments. In [4] after presenting the comparison between different mutations for immune network and clonal selection, performance of these algorithms has been compared with them.

Mutation in genetic algorithm is used for preventing premature convergence by restoring unseen or unexplored solution into the population. But in the AIS, mutation acts probability as only and most important operator that called Hypermutation, in which probability of mutation (P_m) is

978-1-4244-7375-5/10/$26.00 2010 © IEEE

proportional to the affinity between immune cells. A part of the immune cells with high affinity value suffers the lowest mutation rate, whereas the lowest affinity immune cells have high mutation rate. Small value of Pm in genetic algorithm is necessary for working successfully, because it increases diversity of immune cells and escape from local optimum. Moreover, while increasing the generations, Pm must be decreased to maintenance the memory cells with high objective function value. Therefore in AIS algorithms, mutation should be act effectively as only and the most important operator. Generally most of mutation methods use Gaussian or uniform distribution. Besides [4] reported and tested seven types of mutation operators for AIS. Which six of them are designed for real value representation and one of them as binary representation. The authors discussed about parameter adaptation in most of mutation methods. So the control parameter of mutation must be justified for each landscape.

III. LEARNING AUTOMATA

A learning automaton [7,12,16-18] is an adaptive decision-making unit that improves its performance by learning how choose the optimal action from a finite set of allowed actions through repeated interactions with a random environment. The action is chosen at random based on a probability distribution kept over the action-set and at each instant the given action is served as the input to the random environment. The environment responds the taken action in turn with a reinforcement signal. The action probability vector is updated based on the reinforcement feedback from the environment. The objective of a learning automaton is to find the optimal action from the action-set so that the average penalty received from the environment is minimized.

The environment can be described by a triple $E \equiv \{\alpha, \beta, c\}$, where $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ represents the finite set of the inputs, $\beta \equiv \{\beta_1, \beta_2, ..., \beta_m\}$ denotes the set of the values can be taken by the reinforcement signal, and $c \equiv \{c_1, c_2, ..., c_r\}$ denotes the set of the penalty probabilities, where the element c_i is associated with the given action α_i. If the penalty probabilities are constant, the random environment is said to be a stationary random environment, and if they vary with time, the environment is called a non stationary environment. The environments depending on the nature of the reinforcement signal β can be classified into P-model, Q-model and S-model. The environments in which the reinforcement signal can only take two binary values 0 and 1 are referred to as P-model environments. Another class of the environment allows a finite number of the values in the interval [0, 1] can be taken by the reinforcement signal. Such an environment is referred to as Q-model environment. In S-model environments, the reinforcement signal lies in the interval [a, b].

Learning automata can be classified into two main families [18]: fixed structure learning automata and variable structure learning automata. Variable structure learning automata are represented by a triple $< \underline{\beta}, \underline{\alpha}, T >$, where $\underline{\beta}$ is the set of inputs, $\underline{\alpha}$ is the set of actions, and T is learning algorithm. The learning algorithm is a recurrence relation which is used to modify the action probability vector. Let $\alpha(k)$ and $p(k)$ denote the action chosen at instant k and the action probability vector on which the chosen action is based, respectively. The recurrence equation shown by (1) and (2) is a linear learning algorithm by which the action probability vector $\underline{p}$ is updated. Let $\alpha_i(k)$ be the action chosen by the automaton at instant k.

$$p_j(n+1) = \begin{cases} p_j(n) + a \times [1 - p_j(n)] & i = j \\ (1-a) p_j(n) \quad \forall_j & i \neq j \end{cases} \tag{1}$$

When the taken action is rewarded by the environment (i.e. $\beta(n) = 0$) and

$$p_j(n+1) = \begin{cases} (1-b) p_j(n) & i = j \\ \left(\frac{b}{r-1}\right) + (1-b) p_j(n) \quad \forall_j & i \neq j \end{cases} \tag{2}$$

When the taken action is penalized by the environment (i.e. $\beta(n) = 1$). r is the number of actions which can be chosen by the automaton, $a(k)$ and $b(k)$ denote the reward and penalty parameters and determine the amount of increases and decreases of the action probabilities, respectively. If $a(k) = b(k)$, the recurrence equations (1) and (2) are called linear reward-penalty (L_{R-P}) algorithm, if $a(k) >> b(k)$ the given equations are called linear reward-ε penalty ($L_{R-\varepsilon P}$), and finally if $b(k) = 0$ they are called linear reward-inaction (L_{R-I}). In the latter case, the action probability vectors remain unchanged when the taken action is penalized by the environment. In the multicast routing algorithm presented in this paper, each learning automaton uses a linear reward-inaction learning algorithm to update its action probability vector.

IV. THE ADAPTIVE MUTATION OPERATOR BASED ON LA

In AIN, mutation, as the only and most important operator, should be acted effectively, and it is also termed hyper mutation. The immune cells in immune algorithm suffer the mutation according the probability of mutation rate, calculated by a Gaussian distribution following equation (3):

$$c' = c + \alpha \times N(0,1),$$
$$\alpha = \frac{1}{\beta} \times exp(-f^*) \tag{3}$$

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 for control the inverse exponential function and f* is the fitness of an individual normalized [19].

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 have

high mutation rate. Probability of mutation is adaptively assigned by learning automata.

Therefore three actions are considered for LA, namely "*increasing Pm*", "*decreasing Pm*" and "*fixing Pm*" are considered for LA. 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. In every step, LA selects an action according to the probabilistic vector and thus the Pm becomes modified. Consequently, immune cells are mutated by new probability of mutation. Afterwards, based on evaluating the performance, the probabilistic vector of action becomes up to date. The general structure of LA has been shown in figure 1.

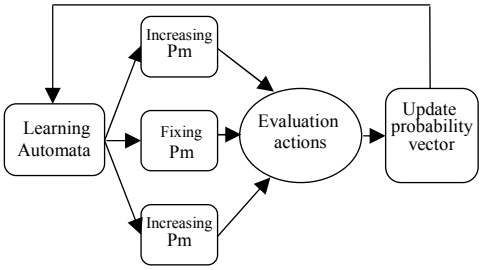

Figure 1. The structure of the proposed learning automata for assign adaptive probability of mutation

At the end of each step, based on the density between immune cells, some parts of immune cells 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].

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

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 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.

According to the description above, the algorithm using proposed adaptive mutation for dynamic environments is illustrated in figure 2.

If there is not 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 will be evaluated as positive, and if the changes are insignificant, it is not logical to change the probabilistic vector of automata, otherwise it is evaluated as negative.

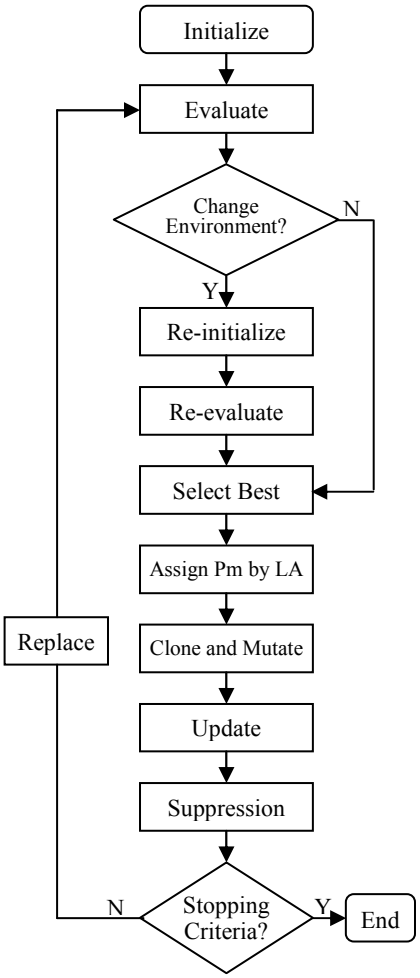

Figure 2. Proposed algorithm with adaptive mutation based on LA

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

V. SIMULATION RESULTS

DF1 [20] is one of the well-known dynamic function generators used in the optimization problems to create dynamic functions. It consists of some cones whose location, heights and slopes are changed over time. The DF1 test function generator proposed by Morrison and De Jong, that is used here to employ the experiments. In this function generator the dynamic objective function is defined as:

$$f(x,y) = \max_{i=1,N} \left[H_i - R_i \times \sqrt{(x - x_i)^2 + (y - y_i)^2} \right] \qquad (5)$$

Where N specifies the number of cones in the search space. Each cone is specified by its location (x_i ,y_i), height H , and slope R with ranges: $H_i \in [H_{base}, H_{base} + H_{range}]$, $R_i \in [R_{base}, R_{base} + R_{range}]$, $x_i \in [-1, 1]$, and $y_i \in [-1, 1]$.
Therefore, dynamic environments can be obtained by specifying the above parameters, and the function complexity depends on the number of peaks, as shown in Figure 3.

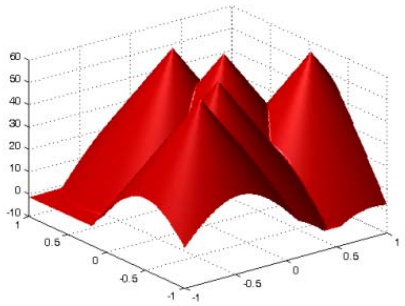

Figure 3. DF1 function

The role of A as a coefficient within [1, 4] in DF1 is to progressively produce more complexity behaviors [20].
In the set of experiments, the objective function consists of 5 and 10 cones with constant H and R and changing location. According to [21], the search space is a 2-D environment with the range [-1, 1]. To create the dynamic objective function, H and R are respectively set to $H \in [3, 6]$ and $R \in [2, 7]$.
In order to evaluate the efficiency of the results of the algorithm to tracking the global optima of DF1 under changing situations, the offline error is evaluated as each generation[21-23].The global error (e_{gt}) is the best fitness compared with the global optimum:

$$e_{gt} = 1 - \frac{f_{gt}}{g_{gt}} \qquad (6)$$

Where f_{gt} is the best fitness at t, and g_{gt} is the actual global optimum. The best e'_{gt} since the previous change occurring in the environment is calculated as:

$$e'_{gt} = \min\{e_{g\tau}, e_{g\tau+1}, \ldots, e_{gt}\} \qquad (7)$$

Where τ is the last iteration, at which the environment changes. Hence, the offline error $e'_{g(offline)}$ is defined as:

$$e'_{g(offline)} = \frac{1}{T}\sum_{t=1}^{T} e'_{gt} \qquad (8)$$

Where T is the number of running iterations.
The performance results of proposed algorithms based on AIN using adaptive mutation as (LA-AIN) in comparison with Ant Colony Optimization as (ACO) [24], Clonal Selection as (CSA) [24], and Hybrid Optimization Algorithm based on Ant Colony Optimization and Immune Principle as (HACO-IA) [21] are given in tables of 1 to 3.

TABLE I. PERFORMANCE COMPARISON OF THE ALGORITHMS IN DF1 WITH N=10, F=1 (OPTIMA±STD)

	A		
Algorithm	**1.5**	**2.5**	**3.5**
ACO	0.076±0.008	0.057±0.013	0.089±0.008
CSA	0.078±0.050	0.067±0.006	0.062±0.007
HACO-IA	0.035±0.007	0.032±0.013	0.050±0.013
LA-AIN	0.055±0.014	0.025±0.013	0.048±0.010

TABLE II. PERFORMANCE COMPARISON OF THE ALGORITHMS IN DF1 WITH N=10, F=10 (OPTIMA±STD)

	A		
Algorithm	**1.5**	**2.5**	**3.5**
ACO	0.015±0.003	0.020±0.003	0.045±0.012
CSA	0.020±0.003	0.025±0.006	0.039±0.003
HACO-IA	0.016±0.004	0.021±0.005	0.019±0.002
LA-AIN	0.013±0.011	0.020±0.014	0.017±0.012

TABLE III. PERFORMANCE COMPARISON OF THE ALGORITHMS IN DF1 WITH N=10, F=100 (OPTIMA±STD)

	A		
Algorithm	**1.5**	**2.5**	**3.5**
ACO	0.005±0.001	0.011±0.001	0.005±0.001
CSA	0.015±0.002	0.011±0.004	0.011±0.002
HACO-IA	0.001±0.001	0.001±0.001	0.002±0.001
LA-AIN	0.014±0.008	0.010±0.003	0.003±0.005

The behaviors of the offline error versus generations for algorithms in the case of four cones with A =1, and changing frequencies (f=1 and 10) are demonstrated in figure 7 and 8 respectively.

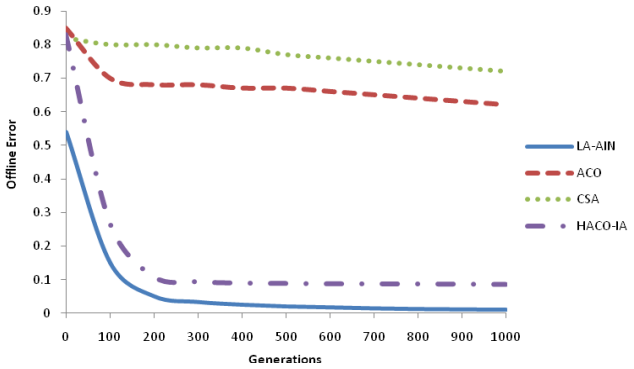

Figure 4. Offline error vs. generations of algorithms in DF1 (f=1, A=1)

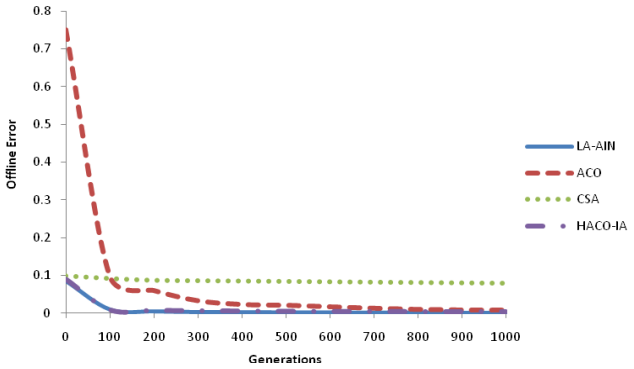

Figure 5. Offline error vs. generations of algorithms in DF1 (f=10, A=1)

Performance result of tables and figures indicate that the proposed algorithm is capable of providing acceptable performance in most of dynamic cases. The success of the proposed algorithm is mainly result from adaptation in mutation using LA, because LA tries to find optimum value for probability of mutation to enhance its search in order to increase diversity of population. Additionally the main advantage of proposed algorithm, no parameter tuning was essential in the algorithm in each search space.

VI. CONCLUSIONS

This paper has been proposed a hybrid algorithm using AIN and LA. In AIN every immune cell equipped with LA. In dynamic environment after detecting the environmental changes there is not guarantee for the exact rate of Pm rate. In the proposed algorithm, in order to improve the diversity of population and enhance the exploration, LA tries to find optimum value of Pm rate adaptively. Simulation results on dynamic functions, are compared with ACO, CSA and hybrid a hybrid algorithm based on ACO and CSA. The experiments in this paper demonstrate that in most of cases of dynamic environments, the proposed algorithm can be successfully tracing the optima in the dynamic environments. Considering the main advantage of the proposed algorithm in this paper, no parameter tuning was essential in the algorithm.

ACKNOWLEDGMENT

The authors would like to thank Miss. Safiarian for some improvements.

REFERENCES

[1] R.I. Lung and D. Dumitrescu, "Evolutionary Swarm Cooperative Optimization in Dynamic Environments," Natural Computing, vol. 9, 2010, pp. 83-94.

[2] 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, vol. 24, 2009, pp. 844-861.

[3] 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, vol. 13, 2009, pp. 763-780.

[4] K. Trojanowski and S.T. Wierzchon, "Immune-based Algorithms for Dynamic Optimization," Information Sciences, vol. 179, 2009, pp. 1495-1515.

[5] S. Yang and X. Yao, "Population-Based Incremental Learning With Associative Memory for Dynamic Environments," IEEE Transactions on Evolutionary Computation, vol. 12, 2008, pp. 542-561.

[6] C. Hu, B. Wang, and Y. Wang, "Multi-Swarm Particle Swarm Optimiser with Cauchy Mutation for Dynamic Optimisation Problems," International Journal of Innovative Computing and Applications, vol. 2, 2009, pp. 123-132.

[7] A. Rezvanian, M.R. Meybodi, "Tracking Extrema in Dynamic Environments using a Learning Automata based Immune Algorithm", Lecture Notes in Computer Science, Lecture Notes in Communications in Computer and Information Science, Proc. International Conference on Control and Automation (CA 2010), 2010 (to appear).

[8] F.O. De Franca, F.J. Von Zuben, and L.N. De Castro, "An Artificial Immune Network for Multimodal Function Optimization on Dynamic Environments," Proc. 2005 Conference on Genetic and Evolutionary Computation, IEEE Press, 2005, pp. 289-296.

[9] S. Xuhua and Q. Feng, "An Optimization Algorithm Based on Multi-Population Artificial Immune Network," Proc. 5th International Conference on Natural Computation, IEEE Press 2009, pp. 379-383.

[10] L.N. De Castro and F. Von Zuben, "Learning and Optimization using the Clonal Selection Principle," IEEE Transactions on Evolutionary Computation, vol. 6, 2002, pp. 239-251.

[11] J. Timmis, A. Hone, T. Stibor and E. Clark, "Theoretical Advances in Artificial Immune Systems," Theoretical Computer Science, vol. 403, 2008, pp. 11-32.

[12] A. Rezvanian and M.R. Meybodi, "LACAIS: Learning Automata based Cooperative Artificial Immune System for Function Optimization,", Lecture Notes in Communications in Computer and Information Science, Contemporary Computing, vol. 94, 2010, pp. 64-75.

[13] F. Abtahi, M.R. Meybodi, M.M. Ebadzadeh, and R. Maani, "Learning Automata-based Co-evolutionary Genetic Algorithms for Function Optimization," Proc. 6th International Symposium on Intelligent Systems and Informatics, IEEE Press, 2008, pp. 1-5.

[14] M. Hamidi and M.R. Meybodi, "New Learning Automata based Particle Swarm Optimization Algorithms," Proc. 2nd Iranian Data Mining Conference, 2008, pp. 1-15.

[15] K. Verbeeck and A. Nowe, "Colonies of Learning Automata," IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 32, 2002, pp. 772-780.

[16] J.A. Torkestani and M.R. Meybodi, "Weighted Steiner Connected Dominating Set and its Application to Multicast Routing in Wireless MANETs," Wireless Personal Communications, 2010, pp. 1-25.

[17] H. Beigy and M.R. Meybodi, "Asynchronous Cellular Learning Automata," Automatica, vol. 44, 2008, pp. 1350–1357.

[18] K.S. Narendra and M.A.L. Thathachar, "Learning Automata: An Introduction. 1989," Printice-Hall, New York.

[19] D. Yongshou, L. Yuanyuan, W. Lei, W. Junling, and Z. Deling, "Adaptive Immune-Genetic Algorithm for Global Optimization to Multivariable Function," Journal of Systems Engineering and Electronics, vol. 18, 2007, pp. 655-660.

[20] R.W. Morrison and K.A. De Jong, "A Test Problem Generator for Non-Stationary Environments," Proc. International Congress on Evolutionary Computation, 1999, pp. 2047–2053.

[21] X. Wang, X.Z. Gao, and S.J. Ovaska, "A Hybrid Optimization Algorithm based on Ant Colony and Immune Principles," International Journal of Computer Science & Applications, vol. 4, 2007, pp. 30-44.

[22] J. Branke, Evolutionary Optimization in Dynamic Environments, Norwell, MA: Kluwer, 2002.

[23] D. Parrott and X. Li, "Locating and Tracking Multiple Dynamic Optima by a Particle Swarm Model using Speciation," IEEE Transactions on Evolutionary Computation, vol. 10, 2006, pp. 440-458.

[24] X. Wang, X.Z. Gao, and S.J. Ovaska, "An Immune-based Ant Colony Algorithm for Static and Dynamic Optimization," IEEE International Conference on Systems, Man and Cybernetics, 2007, pp. 1249-1255.