Solving Maximum Clique Problem in Stochastic Graphs Using Learning Automata

Mohammad Soleimani-Pouri
Department of Electrical, Computer & Biomedical Engineering, Qazvin branch, Islamic Azad University, Qazvin, Iran
m.soleimani@qiau.ac.ir

Alireza Rezvanian
Computer Engineering & Information Technology Department, Amirkabir University of Technology, Tehran, Iran
a.rezvanian@aut.ac.ir

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

***Abstract*—The maximum clique of a given graph *G* is the sub-graph *C* of *G* such that two vertices in *C* are adjacent in *G* with maximum cardinality. Finding the maximum clique in an arbitrary graph is an NP-Hard problem, motivated by the social networks analysis. In the real world applications, the nature of interaction between nodes is stochastic and the probability distribution function of the vertex weight is unknown. In this paper a learning automata-based algorithm is proposed for solving maximum clique problem in the stochastic graph. The simulation results on stochastic graph demonstrate that the proposed algorithm outperforms standard sampling method in terms of the number of samplings taken by algorithm.**

Keywords- maximum clique problem; NP-Hard; stochastic graph; learning automata; social networks.

I. INTRODUCTION

Let $G=(V,E)$ be an undirected graph with vertex set $V=\{1, 2, \ldots, n\}$ and edge set $E \subseteq V \times V$, A clique [1-4] of G is the set of vertices $C \subseteq V$, such that $i,j \in E$ for all $i,j \in C$. A maximum clique is a clique with maximum cardinality among all cliques of G. Due to its numerous applications, the maximum clique problem is one of the most important NP-hard problems [5] and it has been extensively studied in the literature such as clustering in social networks [6-7]. One of the most interests in the networks applications is finding dense subsets of vertices such as clique, which represents a group of entities or people, any two of which have a certain type of relationship with each other in social networks [8-9].

In all existing methods for solving maximum clique, it is assumed that the graph is deterministic and thus the weight of its vertices fixed. But in the real world application, this assumption does not hold true, for example availability of users as nodes in the social networks or activities of routers in communication networks is varying over time. So, in this paper, the maximum clique problem in stochastic graphs is introduced, and then a learning automata-based algorithm is proposed for solving this problem, when the probability distribution function of the weight of the vertices is unknown. To evaluate the performance of the proposed algorithm, the number of samples needs to be taken by it from the vertices of the stochastic graph is compared to that of the standard sampling method. According to the simulation results the proposed algorithm in terms of the number of samples taken from stochastic graph is acceptable. The rest of this paper is organized as follows. In the section II, learning automata is introduced. Stochastic graph is described in section III. In section IV, the proposed algorithm based on learning automata for solving maximum clique in stochastic graph is presented. The performance of the proposed algorithm is evaluated through the simulation in section V. Finally section VI concludes the paper.

II. LEARNING AUTOMATA

A learning automaton [10-13] 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 $\alpha \equiv \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ represents the finite set of the inputs, $\beta \equiv \{\beta_1, \beta_2, \ldots, \beta_m\}$ denotes the set of the values can be taken by the reinforcement signal, and $c \equiv \{c_1, c_2, \ldots, c_r\}$ 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: fixed structure learning automata and variable structure learning automata. Variable structure learning automata are represented by a triple $\langle\beta, \alpha, T\rangle$, where β is the set of inputs, Q 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 $a(k)$ 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.

$$\begin{aligned} p_i(n+1) &= p_i(n) + a[1-p_i(n)] \\ p_j(n+1) &= (1-a)p_j(n) \qquad \forall j, \quad j \neq i \end{aligned} \tag{1}$$

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

$$\begin{aligned} p_i(n+1) &= (1-b)p_i(n) \\ p_j(n+1) &= \frac{b}{r-1} + (1-b)p_j(n) \qquad \forall j, \quad j \neq i \end{aligned} \tag{2}$$

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 and b denote the reward and penalty parameters and determine the amount of increases and decreases of the action probabilities, respectively. If a=b, the recurrence equations (1) and (2) are called linear reward-penalty (L_{R-P}) algorithm, if a>>b the given equations are called linear reward-ε-penalty ($L_{R-\varepsilon P}$), and finally if b=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.

III. STOCHASTIC GRAPH

As mentioned, in the most scenarios of network applications, the weight of the vertices/edges of graph is assumed to be fixed, but in real world applications this is not always true and it varies with time. So, we introduce stochastic graph [14-16] for modeling the real networks applications.

A stochastic graph G can be defined by a triple $G = \langle \boldsymbol{V}, \boldsymbol{E}, F\rangle$, where $V = \{v_1, v_2, ..., v_n\}$ denotes the set of vertices, $\boldsymbol{E} \subset \boldsymbol{V}\times\boldsymbol{V} = \{e_1, e_2, ..., e_m\}$ is the edge set, and $F=\{f_1, f_2, ..., f_n\}$ is the probability distribution describing the statistics of vertex weights. In particular, weight w_i of vertex v_i is assumed to be positive random variable with f_i as its probability density function, which is supposed to be unknown for the proposed algorithm. In stochastic graph G, an maximum clique $\phi_i \subset V$ with weight of $W(v_i)$ vertices and expected weight of $\bar{W}(\phi_i)$ defined as $\Phi=\{\phi_1, \phi_2, ..., \phi_r\}$ the set of all its maximum clique such that for all arbitrary vertices of $v_i,v_j \in \gamma_i$, v_i and v_j are adjacent. The maximum clique is defined as a clique with maximum expected weight. In other word, stochastic maximum clique ϕ^* specifies as follows:

$$\phi^* = \underset{\forall \phi_i \in \Phi}{\arg\max}\left\{\bar{W}(\phi_i)\right\} \tag{3}$$

Where $\bar{W}(\phi_i)$ is the expected weight of the clique ϕ_i and the defined as below

$$\bar{W}(\phi_i) = \frac{\sum_{v_i \in \phi_i} \bar{W}(v_i)}{|\phi_i|} \tag{4}$$

Where $\bar{W}(v_i)$ denotes the expected weight of vertex v_i and $|\phi_i|$ is the clique size. Therefore, the stochastic maximum clique of a given stochastic graph G is defined as the stochastic clique with the maximum expected weight.

IV. PROPOSED ALGORITHM

In this section, the proposed algorithm based on learning automata is described for solving the maximum clique problem in stochastic graphs. In this paper, weight of the vertices of graph is assumed to be positive random variable and the parameters of the underlying probability distribution of the vertex weight are unknown. Therefore it is required to estimate the parameters by a statistical method. In the proposed algorithm, each vertices of graph, equipped with a learning automaton, indeed, a network of learning automata isomorphic to the stochastic graph. In this case, the network of automata can be formulated by a triple <$\underline{A}$, α, C>, where $\underline{A}$ denotes the set of the learning automata, α is the set of actions in which α_i specifies the set of actions can be taken by learning automata A_i, for each $\alpha_i \in \alpha$, and $W=\{w_1,...,w_n\}$ is the set of weights such that w_i ($\forall_{i\in\{1, 2, ..., n\}}$) is the random weight associated with automata A_i. The action set of each learning automata v_i equals to its adjacent vertices of v_i. So, the learning automaton assigned to each vertex v_i of the stochastic graph, referred to as α_i, has $n_i=(d_i-1)$ actions such that $\alpha_i=\{\alpha_1, ..., \alpha_{i-1}, \alpha_{i+1}, ..., \alpha_{ni}\}$. At each step of the algorithm, after sampling from some vertices and compute the expected weight of vertices, the candidate maximum clique is constructed by cooperation of learning automata. The learning automata iteratively construct the candidate maximum clique and update the action probability vectors until they find a near optimal solution to the maximum clique problem. The detail of the proposed algorithm is depicted as follow.

In the initialization, a learning automaton A_i is assigned to each vertices v_i of graph G and action probability vector of them are initialized equal by $1/d_i$. Moreover, the candidate maximum clique consider as empty set.

In the proposed algorithm, the following steps repeated until the stopping criteria are met. In this algorithm, the stopping criteria are considered as predefined total number of iterations and exceed the predefined threshold value of probability vector of the maximum clique as follows:

$$P(\phi^t) = \prod_{v_i \in \phi^t} p(v_i) \tag{5}$$

Where ϕ^t denotes the set of vertices in the candidate maximum clique in the step t, $p(v_i)$ is the probability vector of the v_i.

1. All automata are activated and an automaton was selected randomly as A_i, and added into candidate maximum clique set afterward, all automata nonadjacent of A_i is deactivated. Now, automaton A_i chooses one of its actions according to its action probability vector, and deactivates nonadjacent automaton of A_j. Then, the new selected vertex (j) inserted in the candidate maximum clique set as ϕ^t. This process repeated iteratively until there is no any active automaton.

2. Weight of the candidate maximum clique (ϕ^t) which constructed in the step of t is computed according to equation (6).

$$\bar{w}(\phi^t) = \frac{\sum_{v_i \in \phi^t} \bar{w}(v_i)}{|\phi^t|} \tag{6}$$

Where $\bar{w}(v_i)$ and $\bar{w}(\phi^t)$ are the expected weight of vertex v_i and the expected weight of clique ϕ^t respectively. ϕ^t specifies the vertex set of candidate maximum clique in the step of t and $|\phi^t|$ denotes the cardinality of candidate maximum clique ϕ^t.

3. The candidate maximum clique, which obtained in the step of t in comparison with the best candidate maximum clique up to now is evaluated. If the cardinality of current candidate maximum clique is greater than the cardinality of the all candidate maximum clique that found until now, then all chosen learning automata are rewarded and candidate maximum clique was replaced by current maximum clique, otherwise the chosen learning automata are penalized.

The Best result for maximum clique in the stochastic graph obtained in the end of the algorithm.

V. SIMULATION RESULTS

A. Experimental Study

To evaluate the performance of the proposed algorithm, experiments are accomplished on the following stochastic graphs [15-17], which details of them are listed in table 1 to 2, and are demonstrated in figure 2 to 3. These graph model a real communication networks, which the weight of activity/availability of vertices to be random variables.

TABLE I. PROBABILITY DISTRIBUTION OF GRAPH I

Vertex	Weight	Probability
v_1	{2, 8, 12}	{0.9, 0.08, 0.02}
v_2	{10, 24, 35}	{0.85, 0.12, 0.03}
v_3	{6, 18, 24}	{0.88, 0.1, 0.02}
v_4	{12, 22, 30}	{0.85, 0.11, 0.04}
v_5	{17, 35, 50}	{0.75, 0.2, 0.05}
v_6	{3, 7, 10}	{0.68, 0.25, 0.07}
v_7	{4, 19, 15}	{0.75, 0.14, 0.11}
v_8	{5, 10, 12}	{0.65, 0.23, 0.12}

TABLE II. PROBABILITY DISTRIBUTION OF GRAPH II

Vertex	weight	Probability
v_1	{2, 8, 12}	{0.9, 0.08, 0.02}
v_2	{10, 24, 35}	{0.85, 0.12, 0.03}
v_3	{6, 18, 24}	{0.88, 0.1, 0.02}
v_4	{12, 22, 30}	{0.85, 0.11, 0.04}
v_5	{17, 35, 50}	{0.75, 0.2, 0.05}
v_6	{3, 7, 10}	{0.68, 0.25, 0.07}
v_7	{4, 19, 15}	{0.75, 0.14, 0.11}
v_8	{5, 10, 12}	{0.65, 0.23, 0.12}
v_9	{10, 19, 24}	{0.80, 0.14, 0.06}
v_{10}	{18, 27, 36}	{0.94, 0.05, 0.01}

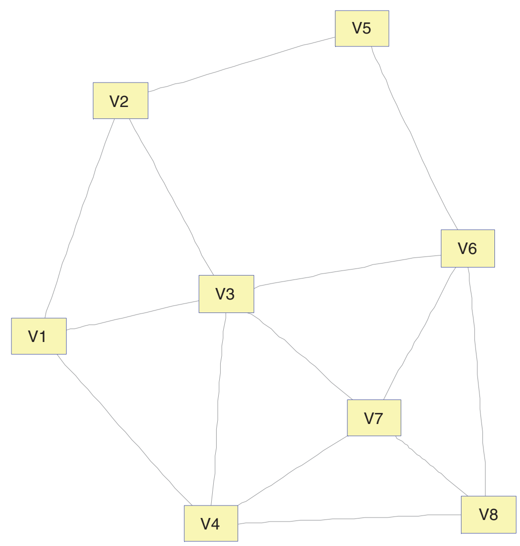

Figure 1. Stochastic graph I

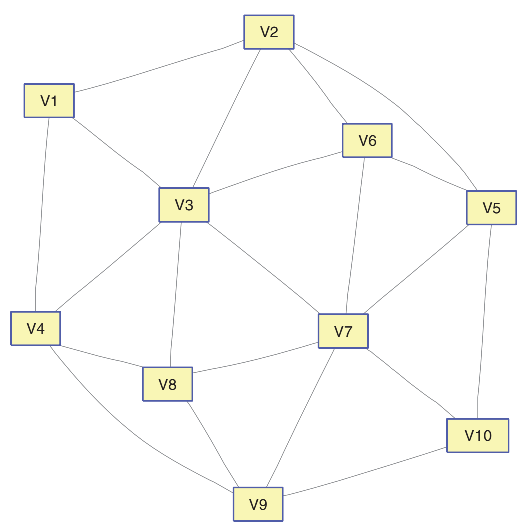

Figure 2. Stochastic graph II

B. Experimental Results

In all simulation presented in this paper, the number of samples taken by the proposed algorithm from the stochastic graph to construct the maximum clique is compared with that of the standard sampling method. The updating scheme for action probability vectors of learning automata is linear reward-inaction (L_{R-I}). The stopping criteria are set to pre-defined number of steps (50000) and threshold value of function on probability vector. The number of samples need to take by standard sampling method with difference confidence level are listed in table 3 to 4 for graph I and II. The proposed algorithm is performed on graphs I and II, and the obtained results in terms of the number of samples taken from the graph are compared with those of the standard sampling method given in tables 5 to 6.

Based on the standard sampling method, to attain a confidence level 1-ε for the maximum clique in stochastic graph, it require to make a confidence level $1\text{-}\varepsilon_i$ for each vertex v_i such that $\Sigma\varepsilon_i=\varepsilon$. It is supposed that the vertices of the stochastic graph have the same confidence level $1\text{-}\varepsilon_0$. So, selecting $\varepsilon_0=\varepsilon/k$, where k is the cardinality of the maximum clique, guarantees a confidence level 1-ε for the maximum clique [15]. The results of the standard sampling method for graphs I and II are listed in the tables 3 and 4, respectively.

TABLE III. AVERAGE SAMPLES TAKEN FROM GRAPH I FOR STANDARD SAMPLING

Vertex	Confidence level				
	0.5	0.6	0.7	0.8	0.9
v_1	317	286	259	243	269
v_2	521	518	541	474	501
v_3	353	337	373	381	353
v_4	406	395	345	418	448
v_5	590	697	642	630	753
v_6	340	265	304	314	315
v_7	311	320	277	315	377
v_8	333	369	377	377	388
Total	3171	3187	3118	3152	3404

TABLE IV. TABLE 4. AVERAGE SAMPLES TAKEN FROM GRAPH II STANDARD SAMPLING

Vertex	Confidence level				
	0.5	0.6	0.7	0.8	0.9
v_1	317	286	259	243	269
v_2	521	518	541	474	501
v_3	353	337	373	381	353
v_4	406	395	345	418	448
v_5	590	697	642	630	753
v_6	340	265	304	314	315
v_7	311	320	277	315	377
v_8	333	369	377	377	388
v_9	242	250	300	312	339
v_{10}	386	425	465	485	454
Total	3802	3867	3887	3953	4201

The results of proposed algorithm for averaged over 30 independent runs in comparison with standard sampling are in terms of the number of samples taken from the stochastic graph are summarized in the table 5 and 6 for graph I and II respectively.

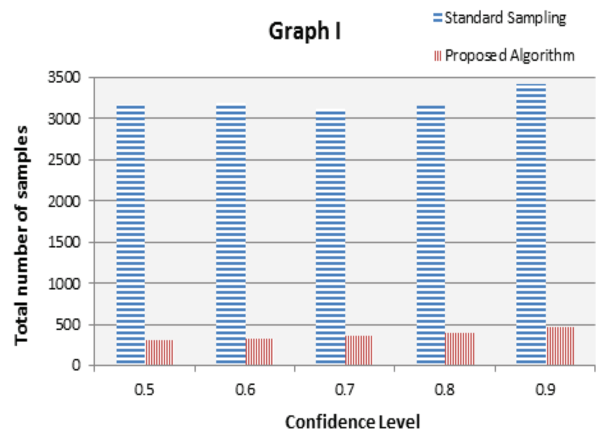

Figure 3. Total number of samples of proposed algorithm and standard sampling for graph I

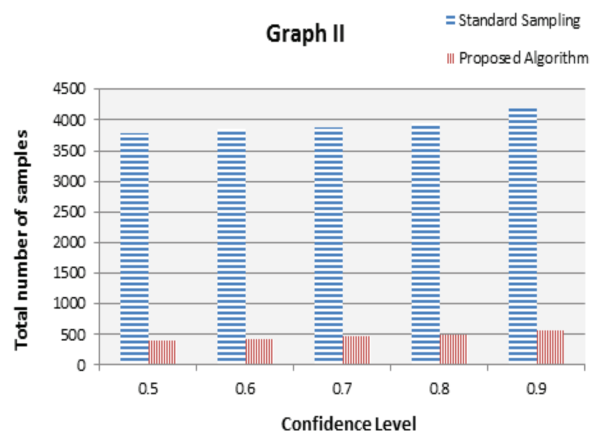

Figure 4. Total number of samples of proposed algorithm and standard sampling for graph II

The simulation results in the figures of 3 and 4 have demonstrated the total number of the total number of samples taken from the stochastic graph for the maximum clique by the proposed algorithm is less than the values obtained by standard sampling method.

In the another experiment, the effect of varying learning parameter for learning automata is evaluated, which listed in the table 5 and 6 for graph I and II respectively.

TABLE V. EFFECT OF LEARNING PARAMETER FOR GRAPH I

Learning parameter	Average clique weight	Average number of samples	Average iterations of algorithm
0.01	9.57	619.69	1062.62
0.02	9.57	432.12	581.13
0.03	9.58	369.15	375.15
0.04	9.59	338.09	266.37
0.05	9.53	318.01	205.85
0.06	9.57	306.76	176.00
0.07	9.57	297.34	163.58
0.08	9.61	289.68	121.78
0.09	9.64	284.87	105.86
0.10	9.56	281.05	86.91

TABLE VI. TABLE 6. EFFECT OF LEARNING PARAMETER FOR GRAPH II

Learning parameter	Average clique weight	Average number of samples	Average iterations of algorithm
0.01	16.05	852.82	3027.98
0.02	16.02	575.73	1491.88
0.03	16.06	483.82	997.59
0.04	16.11	437.18	708.25
0.05	16.05	409.64	562.34
0.06	16.07	390.55	475.36
0.07	16.08	377.86	394.06
0.08	16.05	367.64	332.19
0.09	16.04	360.55	293.01
0.10	16.05	354.27	262.72

The effect of different values of learning parameter in the table 5 and 6 specifies the accuracy of algorithm with increasing the learning parameter in terms of average clique weight, average number of samples, and average iterations of algorithm.

VI. CONCLUSION

In this paper, an algorithm based on learning automata algorithm is proposed to solve the maximum clique in a stochastic graph. Based on the application of real networks, it is assumed that the probability distribution of the vertex weight is unknown. Moreover, in this paper, the stochastic maximum clique was introduced. According to the simulation results, the number of samples taken by the proposed algorithm is less than the standard sampling method for constructing the maximum clique in stochastic graph.

REFERENCES

[1] P. Martins, "Cliques with maximum/minimum edge neighborhood and neighborhood density," *Computers & Operations Research*, vol. 39, no. 3, pp. 594–608, 2012.

[2] Q. Wu, J.-K. Hao, and F. Glover, "Multi-neighborhood tabu search for the maximum weight clique problem," *Annals of Operations Research*, vol. 196, no. 1, pp. 611–634, 2012.

[3] D. Manrique, A. Rodríguez-Patón, and P. Sosík, "On the scalability of biocomputing algorithms: The case of the maximum clique problem," *Theoretical Computer Science*, vol. 412, no. 51, pp. 7075–7086, 2011.

[4] M. Brunato and R. Battiti, "R-EVO: A Reactive Evolutionary Algorithm for the Maximum Clique Problem," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 6, pp. 770 –782, 2011.

[5] R. M. Karp, "Reducibility among combinatorial problems," *Complexity of Computer Computations*, vol. 40, no. 4, pp. 85–103, 1972.

[6] D. Duan, Y. Li, R. Li, and Z. Lu, "Incremental K-clique clustering in dynamic social networks," *Artificial Intelligence Review*, vol. 38, no. 2, pp. 129–147, 2012.

[7] S. Mimaroglu and M. Yagci, "CLICOM: Cliques for combining multiple clusterings," *Expert Systems With Applications*, vol. 39, no. 2, pp. 1889–1901, 2011.

[8] S. Wasserman, *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[9] F. Amiri, N. Yazdani, H. Faili, and A. Rezvanian, "A Novel Community Detection Algorithm for Privacy Preservation in Social Networks," in *Intelligent Informatics*, vol. 18, A. Abraham, Ed. Springer Berlin Heidelberg, 2013, pp. 443–450.

[10] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*. Printice-Hall, 1989.

[11] A. Rezvanian and M. R. Meybodi, "LACAIS: Learning Automata based Cooperative Artificial Immune System for Function Optimization," in *Contemporary Computing*, vol. 94, S. Ranka, Ed. Springer Berlin Heidelberg, 2010, pp. 64–75.

[12] A. Rezvanian and M. R. Meybodi, "Tracking Extrema in Dynamic Environments Using a Learning Automata-Based Immune Algorithm," in *Grid and Distributed Computing, Control and Automation*, vol. 121, T. Kim, Ed. Springer Berlin Heidelberg, 2010, pp. 216–225.

[13] A. Rezvanian and M. R. Meybodi, "An adaptive mutation operator for artificial immune network using learning automata in dynamic environments," in *In: Proceedings of 2010 Second World Congress on Nature and Biologically Inspired Computing (NaBIC),*, Fukuoka, Japan, 2010, pp. 479–483.

[14] J. Akbari Torkestani and M. R. Meybodi, "A learning automata-based heuristic algorithm for solving the minimum spanning tree problem in stochastic graphs," *The Journal of Supercomputing*, vol. 59, no. 2, pp. 1035–1054, 2012.

[15] J. Akbari Torkestani and M. R. Meybodi, "Finding minimum weight connected dominating set in stochastic graph based on learning automata," *Information Sciences*, vol. 200, no. 1, pp. 57–77, 2012.

[16] J. A. Torkestani and M. R. Meybodi, "Learning automata-based algorithms for solving stochastic minimum spanning tree problem," *Applied Soft Computing*, vol. 11, no. 6, pp. 4064–4077, 2011.

[17] H. Beigy and M. R. Meybodi, "Utilizing distributed learning automata to solve stochastic shortest path problems," *International Journal of Uncertainty Fuzziness And Knowledge Based Systems*, vol. 14, no. 5, p. 591, 2006.