Solving the Minimum Spanning Tree Problem in Stochastic Graphs Using Learning Automata

J. Akbari Torkestani
Department of Computer Engineering
Islamic Azad University- Arak Branch
Arak Iran
j-akbari@iau-arak.ac.ir

M. R. Meybodi
Department of Computer Engineering
Amirkabir University
Tehran Iran
mmeybodi@aut.ac.ir

Abstract— **In this paper, we propose some learning automata-based algorithms to solve the minimum spanning tree problem in stochastic graphs when the probability distribution function of the edge's weight is unknown. In these algorithms, at each stage a set of learning automata determines which edges to be sampled. This sampling method may result in decreasing unnecessary samples and hence decreasing the running time of algorithms. The proposed algorithm reduces the number of samples needs to be taken by a sample average approximation method from the edges of the stochastic graph. It is shown that by proper choice of the parameter of the proposed algorithms, the probability that the algorithms find the optimal solution can be made as close to unity as possible.**

Keywords-component; Learning automata, Minimum spanning tree, Stochastic graph

I. INTRODUCTION

A minimum spanning tree (MST) of an edge-weighted network is a spanning tree having the minimum sum of edge weights among all spanning trees. The weight assigned to each edge of the network represents its cost, traversal time, or length depending on the context. MST has several applications, especially in communication networks. In multicast routing protocols, it's one of the most effective methods to multicast the massages from a source node to the destinations. In most scenarios, the edge weights are assumed to be fixed, but this is not always true and they vary with time. For example, the links in a communication network may be affected by collisions, congestions and interferences. Therefore, the MST problem is generalized toward a stochastic MST problem in which the edge weights are not constant but random variables. There have been many studies of the minimum spanning tree problem deal with the deterministic graphs and have been designed some well known algorithms such as Boruvka [1], Kruskal [2] and Prim [3], in which the MST problem can be solved in polynomial time. However, when the edge weights are allowed to be random, the problem becomes considerably more difficult. Unlike the deterministic graphs, the MST problem has not received much attention in study of such stochastic networks.

Ishii et al. [4] proposed a method for solving the stochastic spanning tree problem in which the mentioned problem is transformed into its proxy deterministic equivalent problem and then a polynomial time algorithm presented to solve the latter problem. In this method, the probability distribution of the edges' weights is assumed to be known. Ishii and Nishida [5] considered a stochastic version of the bottleneck spanning tree problem on the edges whose weights are random variables. They showed that, under reasonable restrictions, the problem can be reduced to a minimum bottleneck spanning tree problem in a deterministic case. Mohd [6] proposed a method for stochastic spanning tree problem called interval elimination and introduced several modifications to the algorithm of Ishii et al. [4] and showed that the modified algorithm is able to obtain the better results in less time. Ishii and Matsutomi [7] presented a polynomial time algorithm to solve the problem stated in [4] when the probability distribution of the edges weight is unknown. They applied a statistical approach to a stochastic spanning tree problem and considered a minimax model of the stochastic spanning tree problems with a confidence region of unknown distribution parameters. Alexopoulos and Jacobson [8] proposed some methods to determine the probability distribution of the weight of the MST in stochastic graphs in which the edges have independent discrete random weights, and the probability that a given edge belongs to a minimum spanning tree. Dhamdhere et al. [9] and Swamy and Shmoys [10] formulated the stochastic MST problem as a stochastic optimization problem and proposed some approximation approaches to solve two and multistage stochastic optimization problems.

In this paper, we propose some learning automata-based algorithms to solve the minimum spanning tree problem in a stochastic graph when the probability distribution function of the edge's weight is unknown. The proposed algorithms reduce the number of samples needs to be taken by a sample average approximation method from the edges of the stochastic graph. It is shown that, as the algorithm proceeds, the process of sampling from the graph is concentrated on the edges which form the spanning tree with the minimum expected cost, and so by a proper choice of the parameters of the algorithms, the probability with which the given algorithms find the optimal solution is close enough to unity.

The rest of the paper is organized as follows. The stochastic graphs and the learning automata are described in section II. In section III, the proposed learning automata-based algorithms are presented. Section IV studies the relationship between the learning parameter and the convergence rate of the given algorithms. Section V concludes the paper.

978-0-7695-3595-1/09 $25.00 © 2009 IEEE
DOI 10.1109/ICIME.2009.139

II. LEARNING AUTOMATA AND STOCHASTIC GRAPH

A. Learning Automata

A learning automaton [11-18] is an adaptive decision-making unit that improves its performance by learning how to choose the optimal action from a finite set of allowed actions through repeated interactions with a random environment. 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 $< \underline{\beta}, \underline{\alpha}, T >$, 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 $\underline{p}(k)$ 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.

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

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

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

When the taken action is penalized by the environment (i.e., $\beta(n) = 1$). r is the number of actions 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.

B. Stochastic graph

A An undirected stochastic graph G can be defined by a triple $G = (V, \mathrm{E}, W)$, where $V = \{v_1, v_2, ..., v_n\}$ is the vertex set, $E \subset V \times V$ is the edge set, and matrix $W_{n \times n}$ denotes the probability distributions of the edges weight in stochastic graph G, where n is the number of nodes. For each edge e_{ij} the associated weight is assumed to be a positive random variable with probability density function w_{ij}, which is assumed to be unknown for the proposed algorithms. An undirected stochastic sub graph $G' = (V', \mathrm{E}', W)$ is also called a stochastic spanning tree of G if G' is a connected sub graph, G' has the same vertex set as G, and $\mathrm{E}' \subseteq \mathrm{E}$, satisfying $|E'| = n\text{-}1$, where $|E'|$ denotes the cardinality of set E'. Let $T = \{\tau_1, \tau_2, \tau_3, \ldots\}$ and $\overline{W}\tau_i$ denotes the set of all possible stochastic spanning trees of G as defined above and the expected weight of spanning tree τ_i, respectively. The stochastic MST is defined as a stochastic spanning tree with the minimum expected weight. In other words, a given stochastic minimum spanning tree $\tau^* \in T$ is the stochastic MST if and only if $\overline{W}\tau^* = \min_{\forall \tau_i \in T} \{\overline{W}\tau_i\}$ [4, 7].

III. THE PROPOSED MST ALGORITHMS

In this section, we propose five learning automata-based algorithms to solve the stochastic MST problem which focus on finding the spanning tree with the minimum expected weight in a stochastic graph. At each instant, the automata choose the actions so that the sequence of the chosen actions forms a particular spanning tree in the stochastic graph. The environment responds the selected actions by evaluating the weight sampled from the given spanning tree. The actions which form the spanning tree would be rewarded or penalized depending on whether the environment response is favorable or unfavorable.

Let $G = (V, \mathrm{E}, W)$ denotes the input undirected stochastic graph, where V is the vertex set, E is the edge set, and matrix W denotes the probability distribution of the edge weight in the stochastic graph. A network of learning automata isomorphic to the stochastic graph is formed by assigning to each node of the graph a learning automaton. The resulting network can be described by a triple $< \underline{A}, \underline{\alpha}, W >$, where $\underline{A} = \{A_1, A_2, ..., A_m\}$ denotes the set of the learning automata, $\underline{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_m\}$ denotes the set of actions in which $\alpha_i = \{\alpha_{i1}, \alpha_{i2}, ..., \alpha_{ij}, ..., \alpha_{i\,ri}\}$ defines the set of actions can be taken by learning automata A_i, for each $\alpha_i \in \underline{\alpha}$. Edge $e_{(i,j)}$ corresponds either to the action α_{ij} of the learning automata A_i or to the action α_{ji} of the learning automata A_j. Weight $w_{i,j}$ is the cost associated with edge $e_{(i,j)}$ and assumed to be a positive random variable with an unknown probability distribution. For optimizing the behavior of the learning automata, the proposed algorithm deals with the automata with changing number of actions[11]. Each automaton is initially set in a disabled mode and it is immediately activated after choosing.

Step1. In this step, the learning automata are sequentially and randomly activated and choose one of their actions according to their action probability vectors. After choosing an action by a learning automaton, the disabled automata update their action sets by disabling the actions may form a cycle along the traversed spanning tree.

Step2. The step1 is repeated (or the automata are sequentially activated) until either the number of the selected edges is greater than or equal to $(n-1)$ or there are no more automata which have not already been activated**.**

Step3. The weight of the traversed spanning tree is computed and then compared to the dynamic threshold T_k. At stage $k > 1$, the value of the dynamic threshold T_k is defined as

$$T_k = \frac{1}{r}\sum_{i=1}^{r} T(k_i) \tag{3}$$

Where k_i denotes the number of times the spanning tree τ_i is traversed and $T(k_i)$ denotes the average weight of the spanning tree τ_i and is defined as

$$T(k_i) = \frac{1}{k_i}\sum_{j=1}^{k_i} W\tau_i(j) \tag{4}$$

Step4. Depending on the result of the comparison in the previous step, all the learning automata along the traversed spanning tree update their action probabilities by using the L_{R-I} learning algorithm. To do so, if the weight of the traversed spanning tree is less than or equal to the dynamic threshold, i.e. $W\tau_i(k+1) \le T_k$, the activated automata are rewarded with probability $d_i(k)$, else the activated automata are penalized. The probabilities with which the activated automata are penalized or rewarded vary with time and defined as follows

$$\begin{aligned} c_i(k) &= \text{prob}\ [L\tau_i(k+1) > T_k] \\ d_i(k) &= 1 - c_i(k) \end{aligned} \tag{5}$$

Step5. The process of traversing the spanning trees and updating the action probability vectors are repeated until the product of the probabilities of the edges along the traversed spanning tree is greater than a certain threshold or the number of traversed trees exceeds a pre-specified threshold. The last spanning tree which is traversed before stopping the algorithm is the spanning tree with the minimum expected weight among all spanning trees of the graph. The proposed algorithm is shown by algorithm 1 in more detail.

The automata with fixed action sets may cause some cycles to be formed in traversing along the spanning trees. Therefore, the proposed algorithm deals with the learning automata with changing number of actions[11] for optimizing the behavior of the automata. In this method, the algorithm updates the action probability vectors so that the formation of a cycle be impossible. To implement this approach, the algorithm avoids the cycles by disabling the actions of the automata which are defined as follows: Let $\pi_{i,j}$ defines the paths which connect node v_i to node v_j, $p_j^i(t)$ and $q_j^i(t)$ denotes the probability of choosing the edge $e_{(i,j)}$ and path $\pi_{i,j}$ at stage t, respectively. Now, if the given edge $e_{(i,j)}$ is chosen by the node v_i (or automaton A_i) at stage t, then the edge $e_{(k,i)}$ and its corresponding action will be disabled for all path $\pi_{j,k}$. The given actions are reactivated after updating the action probability vectors of the automata along the traversed spanning tree.

The same and fixed learning parameters give the same importance to all actions and automata which may decrease the convergence rate of the algorithm. This may either lead the algorithm to a premature convergence and so a non-optimal solution for some learning automata or a delayed convergence for some others, while it can be converged in a lesser number of steps. Since a positive random variable is associated with each action of the learning automaton which denotes the action (or edge) cost, it is expected that the sampling rate of the learning algorithm to be related to the probability distribution parameters of the given random variables. Therefore, in the remainder of this section, the effects of the probability distribution parameters on finding an appropriate learning parameter and, consequently, improving the convergence rate and speed of the learning algorithm are considered. The probability distribution functions of the random variables are assumed to be unknown and needs to be obtained by a sampling method at each stage.

The number of samples need to be taken from a given edge is directly related to the variance of the random variable associated with the edge cost. This means that the algorithm is more slowly converged to the true value of the edge cost when the variance of the corresponding random variable is larger. Therefore, the speed of convergence is expected to be increased for the algorithm 1 when the learning parameter of each edge is affected by its variance. In this approach, the learning parameter is calculated by the equation (6), for all edges $e_{(i,j)}$. The Algorithm which is made by this modification in the algorithm 1 is called algorithm 2.

$$a_{e(i,j)}(k) = a \cdot \sigma_{e(i,j)}(k) \tag{6}$$

Where $k_{e(i,j)}$ و $x_{e(i,j)}(k)$ denotes the number of times the edge $e_{(i,j)}$ is traversed and the weight of the edge $e_{(i,j)}$ at stage k. The edges with lower average cost are more likely to participate in the final spanning tree. Therefore, it is not appropriate to give the same importance for rewarding or penalizing the edges in the traversed spanning tree. It is expected that the convergence rate of the algorithm would be improved when the step length of the edges is affected by their means and so the selected edges with lower average cost are more rewarded than the other edges. In this method, the learning parameter is calculated by the equation (7), for all edges $e_{(i,j)}$ Algorithm 1 in which this modification is made is called algorithm 3.

$$a_{e(i,j)}(k) = \frac{a}{\mu_{e(i,j)}(k)} = a \cdot k_{e(i,j)} \Big/ \sum_{k=1}^{k_{e(i,j)}} x_{e(i,j)}(k) \tag{7}$$

As the algorithm proceeds, the probabilities with which the edges of the optimal spanning tree are chosen, progressively increase as those of the other edges decrease. Therefore, we expect that the convergence speed of the learning algorithm increases when the probabilities of the edges affect their step lengths. The Algorithm which is made by this modification in the algorithm 1 is called algorithm 4, and it calculates the learning parameter by the equation (8), for all edges $e_{(i,j)}$.

$$a_{e(i,j)}(k) = a \cdot P_{e(i,j)}(k) \tag{8}$$

By combining the achieved improvements of both algorithm 3 in convergence rate and algorithm 4 in convergence speed, we expect to obtain a more powerful algorithm considering both rate and speed of the convergence for algorithm 1. In this approach, the learning parameter is calculated by the equation (9), for all edges $e_{(i,j)}$, the algorithm which is the combination of algorithms 3 and 4 is called algorithm 5.

$$a_{e(i,j)}(k) = \frac{a \cdot P_{e(i,j)}(k)}{\mu_{e(i,j)}(k)} \tag{9}$$

IV. CONVERGENCE RESULTS

This section is focused on proving the relationship between the learning rate a (of algorithm 1) and the error parameter ε. The goal is to determine the learning rate under which the probability of traversing along (or converging to) the spanning tree with the minimum expected weight is larger than $1-\varepsilon$. To do that, the convergence of the algorithm 1 is proved by theorem1, when the learning automata update their action probability vectors by the reinforcement scheme L_{R-I}. Then, the relationship between the error parameter (i.e. the error parameter involved in the standard sampling method) and the learning rate of the proposed algorithm 1 is concerned. Since the convergence proof is quite lengthy and can not be stated here, the readers are referred to [19]. The method used to prove the convergence follows the method given in [12, 13] to analysis the behavior and prove the convergence of the learning automata operating in the non-stationary environments.

Theorem1. Let $q_i(k)$ be the probability of traversing spanning tree τ_i at stage k in a stochastic graph. If $\underline{q}(k)$ evolves according to algorithm 1, then for each $\varepsilon > 0$, there exists a step length $a^*(\varepsilon) \in (0,1)$ such that for all $a \in (0, a^*)$, we have

$$Prob[\lim_{k\to\infty} q_i(k) = 1] \geq 1-\varepsilon \tag{10}$$

Proof. The proof is quite lengthy and the main steps of the convergence proof are briefly outlined here. This theorem is proved in the following stages. At first, it is proved that, the penalty probabilities of the spanning trees, (i.e, $c_i(k) = prob[L\tau_i > T_k]$) converge to the constant values of the final penalty probabilities, for large enough values of k. Then, it is shown that, the probability of choosing the spanning tree with the minimum expected weight, i.e, $q_i(k)$, is a sub-Martingale process for large values of k, and so the changes in the conditional expectation of $q_i(k)$ is always nonnegative. In other words, if $\tau_{\min}$ be the minimum spanning tree and $q_{\min}(k)$ be the probability of traversing $\tau_{\min}$, at stage k, then $\Delta q_{\min}(k) > 0$, for large values of k. Finally, the convergence of the algorithm 1 to the minimum spanning tree is proved using Martingale convergence theorems and the proof of the theorem1 is completed■.

Theorem2. Let $q_i(k)$ denotes the probability of traversing along the minimum spanning tree τ_i, at stage k, $(1-\varepsilon)$ denotes the probability of converging to the spanning tree τ_i. If $\underline{q}(k)$ is updated by the algorithm 1, then for every error parameter $\varepsilon \in (0,1)$, there exists a learning parameter $a \in (\varepsilon, q)$ so that,

$$\frac{xa}{e^{xa}-1} = \max_{j\neq i}\left(\frac{d_j}{d_i}\right) \tag{11}$$

Where $1-e^{-xq_i} = (1-e^{-x})\cdot(1-\varepsilon)$ and $q_i = [q_i(k) \mid k=0]$.

Proof. Let d_i^* be the final reward probability of τ_i, $d_i(k)$ be the reward probability of τ_i, at stage k, and $\Gamma_i(q)$ be the probability with which the algorithm 1 converges to the unit vector e_i with initial probability vector $\underline{q}$ and defined as

$$\Gamma_i(q) = prob[q_i(\infty)=1 \mid \underline{q}(0)=\underline{q}] = prob[q^* = e_i \mid \underline{q}(0)=\underline{q}] \tag{12}$$

In [12, 13] it has been shown that, $\phi_i[x,q] = \dfrac{e^{-xq_i/a}-1}{e^{-x/a}-1}$ is a sub-regular function and qualifies as a lower bound on $\Gamma_i(q)$, if there exists x under which the equation (13) is satisfied

$$\frac{1}{V[x]} = \max_{j\neq i}\left(\frac{d_j^*}{d_i^*}\right) \tag{13}$$

Where

$$V[u]=\begin{cases}\dfrac{e^{u}-1}{u} & ;\ u\neq 0\\ 1 & ;\ u=0\end{cases} \qquad (14)$$

On the other hand, in [12] it has been proved that, if $d_j/d_i<1$, for all $j\neq i$, then there always exists a $x>0$ such that the equation (13) is satisfied. Therefore, it is concluded that

$$\phi_i[x,q]\le\Gamma_i(q)\le\frac{1-e^{-xq_i}}{1-e^{-x}} \qquad (15)$$

According to the theorem2, we assume that the probability with which the algorithm 1 converges to the spanning tree with the minimum expected weight is $(1-\varepsilon)$, and therefore we have,

$$\Gamma_i(q)\le 1-\varepsilon \qquad (16)$$

From the inequalities (15) and (16) we conclude that,

$$\frac{1-e^{-xq_i}}{1-e^{-x}}=1-\varepsilon \qquad (17)$$

It can be shown that for every error parameter $\varepsilon\in(0,1)$ there exists a x under which the equation (13) is satisfied, and so we have

$$\frac{x^{*}a}{e^{x^{*}a}-1}=\max_{j\neq i}\left(\frac{d_j^{*}}{d_i^{*}}\right) \qquad (18)$$

It is concluded that for every error parameter $\varepsilon\in(0,1)$ there exists a learning parameter $a\in(\varepsilon,q)$ under which the probability with which the algorithm 1 is converged to the spanning tree with the minimum expected weight is greater than $1-\varepsilon$ and the proof of the theorem is completed ■.

V. CONCLUSION

In this paper, five learning automata-based algorithms proposed to solve the minimum spanning tree problem in a stochastic graph when the probability distribution functions of the edges' weight are unknown. As the algorithm proceeds, the process of sampling from the graph is concentrated on the edges by which the minimum spanning tree is formed, and by a proper choice of the parameters for the algorithms, the probability with which the given algorithms find the optimal spanning tree is close enough to unity.

REFERENCES

[1] O. Boruvka, "O jistem Problemu Minimalnım (about a certain minimal problem)," Praca Moravske Prirodovedecke Spolecnosti, 1926, Vol. 3, pp. 37–58.

[2] J. B. Kruskal, "On the Shortest Spanning Sub Tree of a Graph and the Traveling Salesman Problem," Proc. Amer. Math. Soc, 1956, Vol. pp. 748–750.

[3] R. C. Prim, "Shortest Connection Networks and Some Generalizations," Bell. Syst. Tech. Journal, 1957, Vol. 36, pp. 1389–1401.

[4] H. Ishii, S. Shiode, T. Nishida, and Y. Namasuya, "Stochastic Spanning Tree Problem," Discrete Applied Mathematics, Vol. 3, 1981, pp. 263-273.

[5] H. lshii, and T. Nishida, "Stochastic Bottleneck Spanning Tree Problem," Networks, Vol. 13, 1983, pp. 443-449.

[6] I. B. Mohd, "Interval Elimination Method for Stochastic Spanning Tree Problem," Applied Mathematics and Computation, Vol. 66, 1994, pp. 325-341.

[7] H. Ishii, and T. Matsutomi, "Confidence Regional Method of Stochastic Spanning Tree Problem," Mathl. Comput. Modelling Vol. 22, No. 19-12, 1995, pp. 77-82.

[8] C. Alexopoulos, and J. A. Jacobson, "State Space Partition Algorithms for Stochastic Systems with Applications to Minimum Spanning Trees," Networks, Vol. 35, No.2, 2000, pp. 118-138.

[9] K. Dhamdhere, R. Ravi, and M. Singh, "On Two-Stage Stochastic Minimum Spanning Trees," Springer-Verlag Berlin, 2005, pp. 321-334.

[10] C. Swamy, and D. B. Shmoys, "Algorithms Column: Approximation Algorithms for 2-Stage Stochastic Optimization Problems," ACM SIGACT News, 2006, Vol. 37, No. 1, pp. 1-16.

[11] M. A. L. Thathachar and B. R. Harita, "Learning Automata with Changing Number of Actions," IEEE Transactions on Systems, Man, and Cybernetics, 1987, Vol. SMG17, pp. 1095-1100.

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

[13] S. Lakshmivarahan and M. A. L. Thathachar, "Bounds on the Convergence Probabilities of Learning Automata," IEEE Transactions on Systems, Man, and Cybernetics, 1976, Vol. SMC-6, pp. 756-763.

[14] J. Zhou, E. Billard, and S. Lakshmivarahan, "Learning in Multilevel Games with Incomplete Information—Part II," IEEE Transactions on Syustem, Man, and, Cybernetics, 1999, Vol. 29, No.3.

[15] M. A. L. Thathachat, P. S. Sastry, "A Hierarchical System of Learning Automata That Can Learn the Globally Optimal Path," Information Science, 1997, Vol.42, pp.743-766.

[16] M. A. L. Thathachat, V.V.Phansalkar, "Convergence of Teams and Hierarchies of Learning Automata in Connectionist Systems," IEEE Transactions on Systems, Man and Cybernetics, 1995, Vol. 24, pp. 1459-1469.

[17] S. Lakshmivarahan, and K. S. Narendra, "Learning algorithms for two person zero-sum stochastic games with incomplete information: A unified approach," SIAM Journal of Control Optim., 1982, Vol. 20, pp. 541–552.

[18] K. S. Narenrdra, andM. A. L. Thathachar, "On the Behavior of a Learning Automaton in a Changing Environment with Application to Telephone Traffic Routing," IEEE Transactions on Systems, Man, and, Cybernetics,1980, Vol. SMC-l0, No. 5.

[19] J. Akbari, and M. R. Meybodi, "Learning Automata-Based Algorithms for Solving the Minimum Spanning Tree Problem in Stochastic Graphs," Technical Report, Computer Engineering Department, Amirkabir University, Tehran, Iran, June 2008.