International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:7, 2008

Performance Analysis of Learning Automata-Based Routing Algorithms in Sparse Graphs

Z.Farhadpour, Mohammad.R.Meybodi

***Abstract*—**A number of routing algorithms based on learning automata technique have been proposed for communication networks. How ever, there has been little work on the effects of variation of graph scarcity on the performance of these algorithms. In this paper, a comprehensive study is launched to investigate the performance of LASPA, the first learning automata based solution to the dynamic shortest path routing, across different graph structures with varying scarcities. The sensitivity of three main performance parameters of the algorithm, being average number of processed nodes, scanned edges and average time per update, to variation in graph scarcity is reported. Simulation results indicate that the LASPA algorithm can adapt well to the scarcity variation in graph structure and gives much better outputs than the existing dynamic and fixed algorithms in terms of performance criteria.

***Keywords*—**Learning automata, routing, algorithm, sparse graph

I. INTRODUCTION

LEARNING automata (LA) have traditionally been used to model biological learning systems and to learn an optimal action that a Random Environment (RE) offers. Learning is achieved by interacting with the environment, and processing its responses according to the chosen actions. In the learning process an automaton is presented with a set of actions by the environment with which it interacts, and it chooses one of these actions. Base on the chosen action, the automaton is either rewarded or penalized by the environment with a certain probability. Based on this response, the automaton attempts to learn the optimal action. The goal is that the automaton eventually chooses this action more frequently than the other possible actions [8], [9].we provide here a basic introduction to LA, and the way by which it can be used to solve dynamic single source shortest path routing problem. The learning loop (fig.1) involves two entities, the RE and the LA. The actual process of learning is represented as a set of interactions between the RE and the LA. The LA is offered a set of actions $\{\alpha_1, \alpha_2, ..., \alpha_r\}$ by the RE it interacts with, and is limited to choose only one of these actions at any given time. Once the LA decides on an action, α_i, this action will serve as the input to the RE.

Z.Farhadpour is with the Computer Engineering Department, Islamic Azad University, Farahan Branch, Iran (mail: z.farhadpour@gmail.com).

Mohammad.R.Meybodi is with the Computer Engineering Departement, Amirkabir University, Iran (mmeybodi@aut.ac.ir)

The RE will then respond to the input by giving a reward signified by the value "0", or a penalty, signified by the value "1", base on the penalty probability c_i associated with α_i. This response serves as the input to the automaton. Based upon the response from the RE and the current information that it has accumulated so far, the LA decides on its next action and the process repeats.

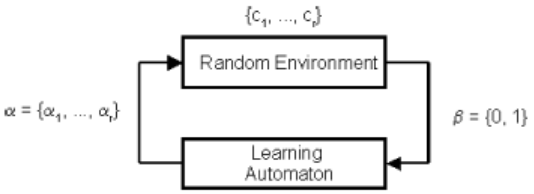

Fig. 1 Automaton–environment feedback loop.

The intention is that the LA learns the optimal action, and eventually chooses this action more frequently than any other action. this paper , uses the family of variable structure stochastic learning automata (VSSA).which are completely defined in terms of action probability updating schemes which are either continuous or discreet. The action probability vector P (n) of an r-action LA is defined as $[P_1(n), P_2(n), ..., P_r(n)]^T$, where $P_i(n)$ is the probability of choosing action α_i at time "n" , with:

$$0 \le P_i(n) \le 1 \qquad \text{and} \qquad \sum_{i=1}^{r} P_i(n) = 1 \qquad (1)$$

Formally, a VSSA can be defined as a quadruple (α, P, β, T), where:

i) $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ is the set of r actions offered by the RE that the LA must choose form.

ii) $P = [P_1(n), P_2(n), ..., P_r(n)]$ is the action probability vector , where P_i represents the probability of choosing action α_i at the n^{th} time instant.

iii) $\beta = \{0,1\}$ is the set of inputs from the RE, where "0" represents a reward and "1" represents a penalty.

iv) T is the updating scheme. This is a map form $P \times \alpha \times \beta$ to P, and defines the method of updating the action probabilities on receiving an input form the RE.

The updating rule that is used here is analogues to the linear-reward-inaction (L_{RI}) scheme, well-known in LA [8].the probability updating schemes for the L_{RI} (whose rationale can be found in [11], [12]) is:

$$\begin{aligned} P_i(n+1) &= 1-\sum_{j\neq i}\lambda_i P_j(n) && \text{if } \alpha_i \text{ is chosen and } \beta = 0 \\ P_j(n+1) &= \lambda_r P_j(n) && \text{if } \alpha_i \text{ is chosen and } \beta = 0 \\ P_j(n+1) &= P_j(n) && \text{if } \beta = 1 \end{aligned} \quad (2)$$

Where λ_r ($0<\lambda_r<1$) is the parameter of scheme. Typically, λ_r is chosen to be close to unity and only rewards are processed inn this scheme.
Given an action probability vector P (t) at time "t", the average penalty is defined as [11]:

$$\begin{aligned} M(t) &= E[\beta(t)\mid P(t)] = \Pr[\beta(t)=1\mid P(t)] \\ &= \sum_{i=1}^{r}\Pr[\beta(t)=1\mid \alpha(t)=\alpha_i].\Pr[\alpha(t)=\alpha_i] \\ &= \sum_{i=1}^{r} c_i P_i(t) \end{aligned} \quad (3)$$

As $t\to\infty$, if the average penalty $M(t)<M_0$, at least asymptotically, the automaton is generally considered to be better than pure-chance automaton. $E[M(t)]$ is given by [11]:

$$E[M(t)=E\{E[\beta(t)\mid P(t)]\}=E[\beta(t)] \quad (4)$$

A learning automaton is considered $\varepsilon-optimal$ if [11]:

$$\lim_{n\to\infty} E[M(t)] < c_1+\varepsilon \qquad \text{where} \quad \varepsilon>0 \quad (5)$$

The L_{RI} scheme has been proven to be $\varepsilon-optimal$ [11].
Learning is not only considered in single automaton case, but also hierarchies of automata and distributed interconnections of automata are used for solving decision problems in a decentralized fashion. A number of routing algorithms based on LA theory have been proposed for dynamic shortest path routing in communication networks. In these algorithms a network of independent decentralized LA controllers is used to learn and maintain shortest path information in a graph. In this paper a comprehensive study is launched to investigate the performance of LASPA the first learning automata solution to the dynamic shortest path routing, across different graph structures with varying scarcities. The rationale behind this is that full connectivity is not always enforceable in real- world situations. The sensitivity of three main performance parameters of the algorithm, being average number of processed nodes, scanned edges and average time per update, to variation in graph scarcity is reported and the algorithm is compared to some fixed and dynamic algorithms in terms of performance criteria. The remaining sections of this paper is organize as follows: section II presents the first learning automata based algorithm for shortest path routing called LASPA , in section III a performance evaluation study is launched to investigate the sensitivity of LASPA algorithm to variation in graph scarcity. In section IV the results are reported and the algorithm is compared with some fixed and dynamic shortest path algorithms.

II.LASPA FOR SOLVING DSSSP

Routing is the distributed activity of building routing tables, one for each node in the network, which tells incoming data packets which outgoing link to use to continue their travel to their destination node. The problem of computing and maintaining the shortest paths information in a graph with a single source where there are continuous probabilistic updates in edge-weights referred to as the Dynamic Single Source Shortest path Problem (DSSSP). In such an environment, one needs to devise efficial solutions to maintain the shortest path even though there are updates on the structure of the graph without recomputing everything from scratch following each topology update. LA-based solution to DSSSP (such as any LA system) consist of three principle components namely, the automaton, the environment, and the reward-penalty structure. In this case, the "system" would imply a team of LA interacting with the stochastic graph and playing a cooperative game.

A. The Automata

In this algorithm a LA is stated at every node in the graph so as to have invoked a game of automata in a sequential fashion. At every instance, its task is to choose a suitable edge from all the outgoing edges in that node. The intention is that it guesses that this edge belongs to the shortest path tree of the "average" overall graph. It accomplishes this by interacting with the environment. It first chooses an action from its prescribed set of actions. It then requests the environment for the current random edge-weight for the edge it has chosen. The system computes the current shortest path by invoking a dynamic shortest path algorithm such as Ramalingam & Rep' s algorithm (RR) or Frigioni algorithm (FMN) whence the LA determines whether the choice it made should be rewarded or penalized.

B.The environment

The environment consists of the overall dynamically changing graph. In the graph, there are multiple edge-weights which change continuously and stochastically. These changes are based on a distribution that is unknown to the LA, but assumed to be known to the environment. In a religious LA-environment feedback, the environment also supplies a reward/penalty signal to the LA. In the algorithm, this feedback is inferred by the system, after it has invoked either the RR or FMN algorithms.

C.Reward /penalty

Based on the action that the LA has chosen, and the edge-weight that the environment provides, the updated shortest path tree is computed. The effect of this choice is now

determined by comparing the cost with the current "average" shortest paths, and the LA thus infers whether the choice should be rewarded or penalized. The automaton then updates the action probabilities using an appropriate scheme, and the cycle continues. The LASPA algorithm works as follows:

Let G (V, E) denoted a dynamically changing directed graph with a set of V vertices and E edges, which is to be processed by LASPA algorithm. Suppose there is a dynamically changing graph where randomly selected edge-weights change depending on a probabilistic distribution. The goal is to determine the underlying shortest path of the average weights, which the system unaware of it. The general pseudo code of LASPA is shown below:

A. Initialization

1. To begin with, the algorithm obtains a snapshot of the directed graph with each edge having a random weight. This edge-weight is based on the random call for an edge, where each edge-cost has a (different) unknown mean and a variance. The algorithm maintains an action probability vector, $P = \{p_1(n), p_2(n),..., p_r(n)\}$ for *each* node of the graph that contains the probability values for choosing different actions, $\{\alpha_1, \alpha_2,..., \alpha_r\}$ offered by the random environment which are the edges leaving the node. These are modeled as the actions. $p_i(n)$, represents the probability of choosing action α_i at the n^{th} time instance. All elements of the action probability vector of a particular node are initialized to have a value equal to one divided by the out degree of that node. A higher probability value indicates a superior action. In this case, each node has a number of possible outgoing edges. Each possible outgoing edge corresponds to a probable action that can be selected for calculating the shortest path tree. Based on the chosen action, the system does shortest path computations, whence it finds whether the random environment inputs the automaton with $\beta = \{0,1\}$ where 0 represents a reward and 1 a penalty. Let α_i denote the action corresponding to choosing the outgoing edge (x,y) from node x to node y,and let dist(x) be the shortest path distance of x from the source, and w(x,y) the weight of edge (x,y):

$$\beta = 0 \Rightarrow dist(x) + w(x, y) < dist(y)$$
$$\beta = 1 \Rightarrow dist(x) + w(x, y) \geq dist(y) \qquad (6)$$

2. Dijkstra's Algorithm is run once to determine the shortest path edges on the graph snapshot obtained in the first step. Based on this, the action probability vector of each node is updated such that the outgoing edge from a node which is determined to belong to the shortest path edge has an increased probability than before the update.

B. Iterations

3. Then a node is randomly chosen from the current graph. For that node, based on the action probability vector, an edge is chosen as follows. For example, if a node has three outgoing edges (three possible actions) and the action probability vector for that node is {0.3, 0.2, 0.5}, the edge (action) chosen is selected based on this distribution.

4. The Environment is then requested for the correct edge-weight of the edge that is randomly selected in Step 3 above. Since the edge-weights are real numbers, the edge-weight should have increased/decreased. Thus, the current shortest path's tree is calculated using either of the existing edge-weight increase/ decrease algorithms (i.e., either RR or FMN algorithm).

5. The action probability vector for the node whose edge was just selected is updated such that the edge that could now potentially belong to the shortest path's tree has a greater likelihood of being selected than before the update.

6. Steps 3–5 above are repeated a large number of times until the algorithm converge.

III. PERFORMANCE ANALYSIS OF LASPA ALGORITHM

A. Experimental setting

As mentioned in section II the environment which is processed by LASPA algorithm consists of a dynamically changing directed graph with a set of vertices and edges. This graph is said to be sparse if the total number of edges is small computed to the total number of possible edges. Or, more formally, in a sparse graph, M=O (N) where N and M are the number of nodes and edges in the graph respectively. In this section the sensitivity of performance of LASPA to increase in graph scarcity is evaluated. Several experiments were designed to evaluate the performance of the algorithm on several graph topologies with single source, directed edges, positive real edge-weights that are normally distributed and with stochastic edge updates and different degree of scarcity. For the above experiments a random generator is used. Random graph topology generator is a module builds a random directed graph with n nodes, e edges, and positive real edge weights. The edge-weights are randomly generated and are normally distributed. The random graph can be built from either specifying the number of nodes and edges or by specifying the graph scarcity.

B. Performance Metrics

Three metrics which were used for evaluating the performance of the algorithm are listed below:

- Number of scanned edges: This quantity measures the number of edges that are scanned in update operations. The first scanned edge is the edge whose weight is changed. Next, when an edge is scanned to check if it is in the shortest path, its value is incremented.
- Number of processed nodes: This quantity measures the number of nodes that are processed in update operations.
- Time required per update operation: This quantity is the running time required to update weights and to obtain all the shortest paths.

IV. RESULTS

This section reports the results of the experiments that were conducted to examine the performance of LASPA in different random graph topologies. The algorithm is compared with two dynamic routing algorithms RR and FMN. The results of three sets of experiments are summarized below:

- Experiment set1: Here the performance of LASPA is compared with two dynamic shortest path routing algorithm RR and FMN. The RR, FMN, LASPA are implemented on a random graph topology with 50 nodes and 20% scarcity. The edge-weights are random real value having a mean between 1.0 and 5.0 and with variance between 0.1 and 0.9. The value of learning parameter was set to 0.98. The results of this experiment are shown in figures2-4. The results show 75-95% improvement of the performance of LASPA with respect to that of FMN and RR.
- Experiment set2: The second set of experiments was conducted to evaluate the RR, FMN and LASPA algorithms varying in graph structures, specifically, in number of nodes in the graphs and the graph scarcity. The results of the experiment are listed in tables I-III.
- Experiment set3: In this experiment, the sensitivity of the performance of LASPA algorithm to increase in graph scarcity keeping other parameters constant is evaluated. Figures 5, 6 show the variation of average time per update and the average number of processed nodes with the variation in the graph scarcity. It was observed that as scarcity of the graph increases, the average time to update and the average number of processed nodes decrease.

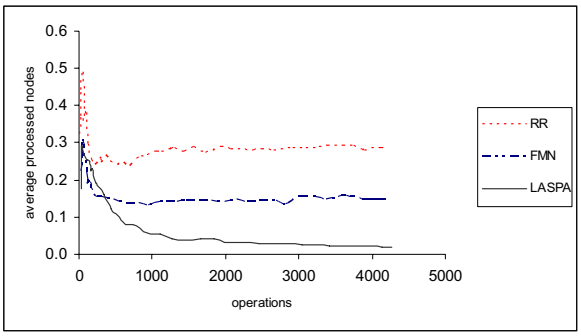

Fig. 2 Plot of the average processed nodes as a function of the number of Operations done for the various algorithms. Here, every "Operation" signifies an increase/decrease of the edge weight, whose inclusion in the shortest path tree is being considered.

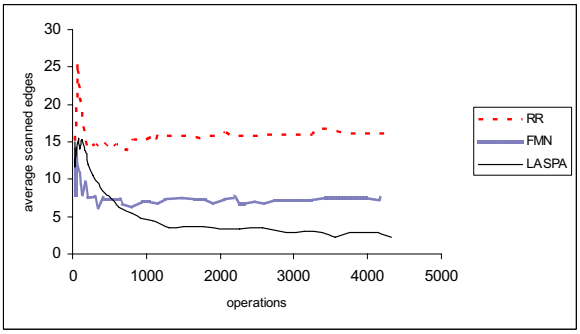

Fig. 3 Plot of the average scanned edges as a function of the number of Operations done for the various algorithms. The term "Operation" has the same significance as in Fig.2.

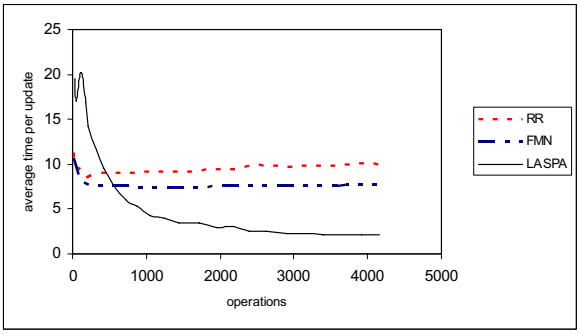

Fig. 4 Plot of the average time per update as a function of the number of Operations done for the various algorithms. The term "Operation" has the same significance as in Fig.2.

TABLE I STATISTICS REPORTING THE NUMBER OF PROCESSED NODES WITH THE VARIATION OF GRAPH SPARSITY

scarcity	RR	FMN	LASPA
10%	0.241	0.251	0.08
30%	0.425	0.196	0.012
50%	0.492	0.216	0.006
70%	0.0872	0.188	0.005
90%	1.174	0.161	0.0036

TABLE II STATISTICS REPORTING THE *NUMBER OF SCANNED EDGES* WITH THE VARIATION OF GRAPH SPARSITY

scarcity	RR	FMN	LASPA
10%	18.0456	11.733	1.0828
30%	19.02	7.113	1.259
50%	16.55	7.119	1.416
70%	16.89	7.574	1.124
90%	16.0	6.944	1.112

TABLE III. STATISTICS REPORTING THE *TIME PER UPDATE* WITH THE VARIATION OF GRAPH SPARSITY

scarcity	RR	FMN	LASPA
10%	6.418	3.732	1.87
30%	6.256	2.534	1.734
50%	6.278	2.606	1.812
70%	4.394	1.954	1.991
90%	3.23	1.674	1.671

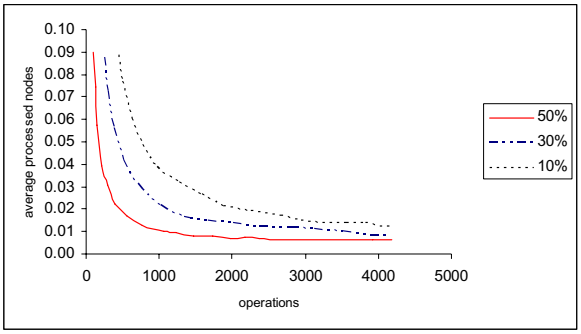

Fig. 5 Sensitivity of average number of processed nodes of LASPA to variation in graph scarcity the term "Operation" has the same significance as in Fig.2.

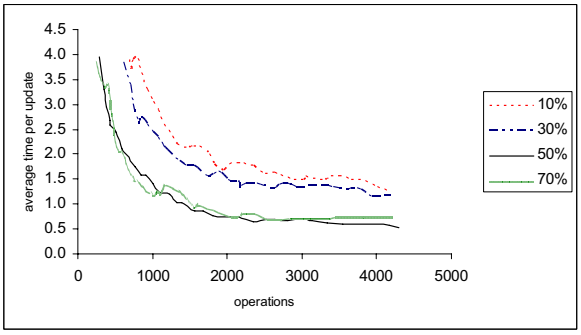

Fig. 6 Sensitivity of average time per update of LASPA to variation in graph scarcity. The term "Operation" has the same significance as in Fig.2.

V. CONCLUSIONS

In this paper, a comprehensive study is launched to investigate the performance of LASPA, the first learning automata based solution to the dynamic shortest path routing, across different graph structures with varying scarcities. The sensitivity of three main performance parameters of the algorithm, being average number of processed nodes, scanned edges and average time per update, to variation in graph scarcity is reported. Simulation results indicate that the LASPA algorithm can adapt well to the scarcity variation in graph structure and gives much better outputs than the existing dynamic and fixed algorithms in terms of performance criteria.

REFERENCES

[1] S. Misra, , B. John Oommen," An Efficient Dynamic Algorithm for MaintainingAll-Pairs Shortest Paths in Stochastic Networks," IEEE TRANSACTIONS ON COMPUTERS, VOL. 55, NO. 6, JUNE 2006.

[2] G. I. Papadimitriou, M. Sklira, and A. S. Pomportsis," A New Class of $\mathcal{E}$ − Optimal Learning Automata," IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, VOL. 34, NO. 1, FEBRUARY 2004.

[3] N. Baba,,Y. Mogami, "A New Learning Algorithm for the Hierarchical Structure Learning Automata Operating in the Nonstationary S-Model Random Environment," IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, VOL. 32, NO. 6, DECEMBER 2002.

[4] M. A. L. Thathachar, P. S. Sastry,"Varieties of Learning Automata: An Overview," IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, VOL. 32, NO. 6, DECEMBER 2002.

[5] M. S. Obaidat, G. I. Papadimitriou, and A. S. Pomportsis, "LearningAutomata: Theory, paradigms and applications," IEEE Trans. Syst., Man,and Cybern., vol. 32, no. 6, pp. 706–709, Dec. 2002.

[6] G. I. Papadimitriou and A. S. Pomportsis, "Learning-automata-based TDMA protocols for broadcast communication systems with burstytraffic," IEEE Commun. Lett., vol. 4, no. 3, pp. 107–109, 2000.

[7] P. Narvaez, K.-Y Siu, and H. Y. Tzeng, "New dynamic algorithms for shortest path tree computation," IEEE/ACM Trans. Networking, vol. 8, pp. 734–746, 2000.

[8] B. J. Oommen and E. V. de St. Croix, "Graph partitioning using learning automata," IEEE Trans. Comput., vol. 45, pp. 195–208, 1995.

[9] B. J. Oommen and T. D. Roberts, "Continuous learning automata solutions to the capacity assignment problem," IEEE Trans. Comput., vol.49, pp. 608–620, Jun. 2000.

[10] G. Ramalingam, Bounded Incremental Computation. Berlin:Springer-Verlag, 1996, vol. 1089, Lecture Notes in Computer Science.

[11] K. S. Narendra and M. A. L. Thathachar, Learning Automata. EnglewoodCliffs, NJ: Prentice-Hall, 1989.

[12] S. Lakshmivarahan, Learning Algorithms Theory and Applications.New York: Springer-Verlag, 1981.