Solving Stochastic Shortest Path Problem Using Monte Carlo Sampling Method: A Distributed Learning Automata Approach

M. R. Meybodi and Hamid Beigy

Soft Computing Laboratory
Computer Engineering Department
Amirkabir University of Technology
Tehran, Iran
meybodi@ce.aku.ac.ir *beigy@ce.aku.ac.ir*

Abstract. In this paper, we introduce a Monte Carlo simulation method based on distributed learning automata (DLA) for solving the stochastic shortest path problem. We give an iterative stochastic algorithm that finds the minimum expected value of set of random variables representing cost of paths in a stochastic graph by taking sufficient samples from them. In the given algorithm, the sample size is determined dynamically as the algorithm proceeds. It is shown that when the total sample size tends to infinity, the proposed algorithm finds the shortest path. In this algorithm, at each instant, DLA determine which edges to be sampled. This reduces the unnecessary sampling from the edges which don't seem to be on the shortest path and thus reduces the overall sampling size. A new method of proof (different from [2,3]) is used to prove the convergence of the proposed algorithm. The simulations conducted confirm the theory.

1 Introduction

The deterministic shortest path problem has been studied extensively and many algorithms reported in the literature. In this problem, one looks for a path joining source and destination nodes while minimizing sum of costs of the traversed edges. However, there are many applications in which cost of edges are random variables [1]. These graphs called *stochastic graphs* and can be defined by a triple $G =< V, E, Q >$, where $V = \{1, 2, \ldots, n\}$ is set of nodes, $E \subset V \times V$ is set of edges, and $n \times n$ matrix Q is the probability distribution describing the statistics of edge costs. In particular, cost C_{ij} of edge (i, j) is assumed to be a random variable with q_{ij} as its probability density function. It is assumed that the distribution q_{ij} is not known a priori.

The path π_i from source node v_s to destination node v_d in the stochastic graph G is defined as an ordering $\pi_i = \{i_1, i_2, \ldots, i_{m_i}\}$ $(\pi_i \subset V)$ in such a way that $i_1 = v_s$ and $i_{m_i} = v_d$ are source and destination nodes, respectively and $(i_j, i_{j+1}) \in E$ for $1 \leq j < m_i$. Assume that there are r paths $\Pi = \{\pi_1, \pi_2, \ldots, \pi_r\}$ between source node v_s and destination node v_d. The shortest path is defined as a path with minimum expected cost. In other

word, the shortest path π^* has the expected cost $C_{\pi^*} = \min_{\pi_i \in \Pi}\{C_{\pi_i}\}$, where $C_{\pi_i} = \sum_{j=1}^{m_i - 1} C_{i_j i_{j+1}}$ and C_{ij} are the expected cost of path π_i and edge (i,j), respectively. It is evident that when enough samples are taken from edges of the graph, the estimated cost of paths approach to their expected value, but the ways to sample, number of samples, and distribution of sampling are important and must be addressed.

DLA is a network of automata which collectively cooperate to solve a particular problem. In DLA, the number of actions for any automaton in the network is equal to the number of outgoing edges from that automaton. When an automaton selects one of its actions, another automaton on the other end of edge corresponding to the selected action will be activated. At any time only one automaton in the network will be active. Formally, a DLA with n learning automata (LA) can be defined by a graph (A, E), where $A = \{A_1, A_2, \cdots, A_n\}$ is the set of automata and $E \subset A \times A$ is the set of edges in the graph in which an edge (i,j) corresponds to action α_j of automaton A_i. Two DLA based algorithms for finding shortest path in stochastic graph are reported in the literature [2,3]. In [2], DLA is introduced and applied to the shortest path problem. In order to compute the probability of path π being the shortest $(p(\pi))$, a DLA is constructed from given graph. Each LA in this DLA updates its action probability vector using L_{R-I} reinforcement scheme [4] until the shortest path is found. In [3], another DLA based algorithm which is faster than the algorithm reported in [2] is given. That is, it requires fewer number of samples taken from the edges of the graph in order to decide which path from source to destination is shortest. The main difference between algorithms reported in [3] and the one reported in [2] is the definition of dynamic threshold. In [2], the dynamic threshold is the average cost of sampled paths whereas in the algorithm given in [3] the dynamic threshold is defined to be the minimum of average cost of sampled paths taken. It has been shown that both algorithms find the shortest path in a stochastic graph with probability as close as to unity.

In this paper, we introduce a Monte Carlo simulation method based on DLA for solving the stochastic shortest path problem. We give an iterative stochastic algorithm that finds the minimum value of set $\{C_{\pi_1}, \ldots, C_{\pi_r}\}$ of random variables representing the expected cost of paths in a stochastic graph by taking sufficient samples from them. In this algorithm, the sample size is determined dynamically as the algorithm proceeds. It is shown that when the total sample size tends to infinity, the proposed algorithm finds the shortest path. In this algorithm, at each instant, DLA determine which edges to be sampled. This reduces the unnecessary sampling from the edges which don't seem to be on the shortest path and thus reduces the overall sampling size. A new method of proof (different from [2,3]) is used to prove that if each LA used in DLA uses the L_{R-I} learning algorithm, the given algorithm finds the shortest path with probability as close as to the unity. The convergence proof reported in [2,3] is based on Martingle theorem whereas the convergence proof

reported in this paper is based on sampling theory. The simulation results show that the proposed algorithm performs better in comparing to algorithms reported in [2,3] for some of the graphs and perform worse for some other.

The rest of the paper is organized as follows: Section 2 presents the proposed Algorithm. Simulation results and discussion are given in section 3 and section 4 concludes the paper.

2 Proposed Algorithm

In this section, we propose an algorithm based on DLA for finding shortest path in a stochastic graph. In this algorithm, the stochastic graph plays the role of random environment for DLA. The output of DLA is a sequence of actions that represent a particular path in the graph. The environment uses the length of this path to produce its response. This response causes the actions along this path be *rewarded* or *penalized*. In the proposed algorithm, at first a network of LA which is isomorphic to the input graph is created. In this network each node is a LA and each outgoing edge of this node is one of the actions of this LA. The algorithm then traverse the graph (as described later) until the shortest path is found. For the sake of simplicity, we use notations shown in figure 1 to describe the algorithm. The action probability vector for automaton A_j is shown by $p^j = (p_1^j, p_2^j, \cdots, p_{r_j}^j)$ where p_m^j denotes the probability of selecting action α_m, that is, edge (j, m). For simplicity in the presentation of algorithm, the actions of an automaton are shown by the indices of adjacent automata. For example, the set of actions of automaton A_2 in figure 1 is represented by $\{\alpha_5, \alpha_6\}$. The variable $AvgCost(k)$ stores the average taken over the cost of the paths traversed up to time k and T denotes the dynamic threshold. Now, we give a general description of the proposed algorithm. In the first step, the source automaton A_s chooses one

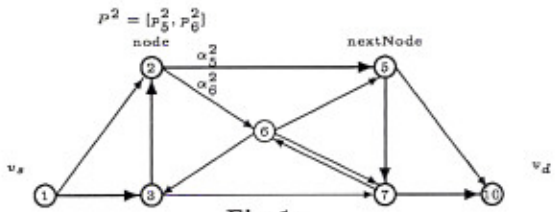

Fig. 1.

of its actions, say α_m. This action activates automaton A_m on the other end of edge (s, m). The process of selection of an action and activating an automaton is repeated until destination automaton A_d is reached or for some reason moving along the edges of the graph is not possible or the number of visited nodes exceeds the number of nodes in the graph. After the A_d is

reached, the length of the traversed path and average cost of sampled paths (*AvgCost*) are computed. Then *AvgCost* is compared with a *dynamic threshold* (described later) and depending on the result of comparison, the value of dynamic threshold is updated. Dynamic threshold T at instant k is the minimum of sequence $\{AvgCost(0), \ldots, AvgCost(k-1)\}$. At the beginning of algorithm, the dynamic threshold is set to average cost of some random generated paths. If *AvgCost* is less than T, then T is set to *AvgCost* and all the selected actions of automata along the traversed path are rewarded. If *AvgCost* is grater than T, then T remains unchanged and all the selected actions of automata along the traversed path are penalized. Updating of action probability vectors are done according to the L_{R-I} algorithm in the direction from A_d to A_s. In updating phase the step lengths of automata at instant k along the traversed path are updated according to the following equation.

$$a_k = \frac{a}{b + \epsilon k} \quad \text{for } k > 1 \text{ and } b \geq 1 > a/\epsilon > 0 \tag{1}$$

The process of travelling from A_s to A_d is repeated until the stopping criteria is reached which at this point the path last travelled has minimum expected length among all the paths from source to destination. The algorithm stops if the product of the probability of choosing the edge of the traversed path, called *path probability*, is greater than a certain threshold. Updating the step length of L_{R-I} scheme according to equation (1) decreases the risk of convergence to non–optimal action [5]. In order to excludes loops from the traversed paths, the algorithm meets every node along a path being traversed at most once. To implement this, if an LA chooses action α_k from the list of its actions, then all unactivated LAs will disable action α_k (but not removed) in their list of actions. When travelling again from source to destination node, all the disabled actions will be enabled.
The main contribution of this paper is summarized by the following theorem.

Theorem 1. If $q(n)$ evolves according to the given algorithm, then the proposed algorithm converges to the shortest path with a probability as close as to unity for graphs with unique shortest path.

Sketch of Proof This theorem is proved in two steps. In the first step, it is shown that every edge in the graph will be sampled infinitely often with probability 1. In the second step, it is first shown that the dynamic threshold is a decreasing function with respect to time and then shown that the dynamic threshold converges to its minimum with probability 1. That is the algorithm finds the path with minimum expected cost with probability as close as to unity. The detailed proof is given in [6].

3 Experiments

In order to study the feasibility of the proposed algorithm, experiments are conducted on stochastic graphs given in figures 3 through 5. These graphs

are borrowed from [7]. Graph 1, which is shown in figure 3 is a graph with 4 nodes, 5 arcs, $v_s = 1$, and $v_d = 4$. Graph 2 that is shown in figure 4 is a graph with 10 nodes, 23 arcs, $v_s = 1$, and $v_d = 10$. Graph 3, which is shown in figure 5 is a graph with 15 nodes, 42 arcs, $v_s = 1$, and $v_d = 15$. To compare the performance of the proposed algorithm and the algorithm reported in [2,3], three sets of experiments are conducted on the above mentioned graphs and the results are summarized in table 1. The proposed algorithm will be terminated when the probability of traversed path is greater than 0.9 or 900,000 paths are traversed. The algorithm is executed 100 times and the results are summarized in two tables. The average number of iterations for converged runs and the percentage of converged runs are given in the first table. The second table for each algorithm, shows the total number of samples taken from all edges in the graph and also the total number of samples taken from the edges along the shortest path. The simulation results show that the proposed algorithm performs better in comparison with algorithms reported in [2,3] for some of the graphs and performs worse for some other graphs. It seems that the graph on which the proposed algorithm is tested plays a major rule on the rate of convergence. For more experimentation refer to [6].

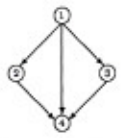

Fig. 3. Graph 1

Fig. 4. Graph 2

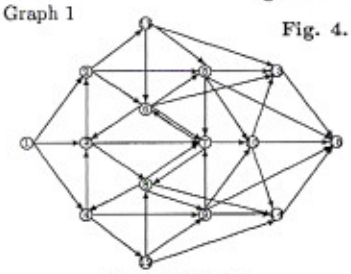

Fig. 5. Graph 3

4 Conclusion

In this paper we have studied the problem of determining the optimal path in a stochastic network. The algorithm presented provides policy which can be used to determine a path from source to destination node with minimal expected cost. This algorithm is an adaptive procedure based on distributed learning automata. A new method of proof (different from [2,3]) is used to

prove the optimality of the proposed algorithm. It seems the graph on which the algorithm is tested plays a major rule on the rate of convergence.

Table 1. The simulation results of Algorithm($\epsilon = 0.01$)

	Graph 1		Graph 2		Graph 3	
a	Avg. Iterations	Runs Converged	Avg. Iterations	Runs Converged	Avg. Iterations	Runs Converged
0.0060	851	100	2400	99	39782	98
0.0061	871	100	2414	100	26325	96
0.0062	847	100	2445	99	26987	98
0.0062	829	100	2253	99	32155	100
0.0063	834	100	2218	99	37585	100
0.0064	833	100	2650	99	33275	98
0.0065	843	100	2475	100	27847	99
0.0066	812	100	2360	99	22688	99
0.0068	810	100	2177	99	20344	97
0.0069	805	100	2149	100	27161	97
0.0070	794	99	2377	98	22996	99

	Graph 1		Graph 2		Graph 3	
a	Avg. Samples	Shortest Path	Avg. Samples	Shortest Path	Avg. Samples	Shortest Path
0.0060	1700	999	9017	4177	141293	35170
0.0061	1740	997	9113	4246	92689	53560
0.0062	1692	978	9074	4307	97396	30580
0.0062	1658	964	8416	3951	114157	28713
0.0063	1666	968	8269	3867	110175	29526
0.0064	1664	960	9614	4465	135480	42219
0.0065	1686	962	9137	4257	99022	25870
0.0066	1624	938	8662	4169	81898	23655
0.0068	1620	931	8075	3795	72277	22654
0.0069	1610	925	8050	3807	95517	30237
0.0070	1572	899	8534	4106	83126	22952

References

1. Polychronopoulos, G. H. and Tsitsiklis, J. N. (1996): Stochastic Shortest Path Problems with Recourse. Newtorks, 27,133–143
2. Meybodi, M. R. and Beigy, H. (2001): Solving Stochastic Shortest Path Problem Using Distributed Learning Automata. Proc. of 6th Annual Int. Computer Society of Iran Computer Conference CSICC-2001, Iran, 70–86
3. Beigy, H. and Meybodi, M. R. (2001): A New Distributed Learning Automata Based Algorithm For Solving Stochastic Shortest Path Problem, Tech. Rep. TR-CE-2001-006, Computer Eng. Dept., Amirkabir Univ. of Tech., Tehran, Iran
4. K. S. Narendra and K. S. Thathachar (1989): Learning Automata: An Introduction, New York: Printice-Hall.
5. Najim, K. and Pozyak, A. S. (1994): Learning Automata: Theory and Applications, Oxford: Pergamon press
6. Meybodi, M. R. and Beigy, H. (2001):A Sampling Method Based on Distributed Learning Automata for Stochastic Shortest Path Problem, Tech. Rep. TR-CE-2001-007, Computer Eng. Dept., Amirkabir Univ. of Tech., Tehran, Iran
7. Alexopoulos, C. (1997): State Space Partitioning Methods for Stochastic Shortest Path Problems, Newtorks, 30, 9–21