A Sampling Method Based on Distributed Learning Automata for Stochastic Shortest Path Problem

M. R. Meybodi

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

Hamid Beigy

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

Abstract: In this paper, we introduce a Monte Carlo simulation method based on distributed learning automata for solving the stochastic shortest path problem. We give an iterative stochastic algorithm that find 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, distributed learning automata 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 [1, 2]) is used to prove the convergence of the proposed algorithm. The simulations conducted confirm the theory.

Keywords: Distributed learning automata, learning automata, shortest path problem, stochastic graph

1 Introduction

The deterministic shortest path problem has been studied extensively and many algorithms have been reported in the literature [3]. 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. 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 ith path π_i from source node v_s to destination node v_d in 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 $1 \leq j < m$. 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 cost $C_{\pi^*} = \min_{\pi_i \in \Pi}\{C_{\pi_i}\}$, where $C_{\pi_i} = \sum_{j=1}^{m_i - 1} C_{i_j i_{j+1}}$ is the cost of path π_i. The stochastic shortest path problem can be formulated on the

basis of time that values of edge costs are learned [4]. For example,

1. The values of edge costs are learned before a path is traversed [5]. This approach has some difficulties such as
   (a) How many samples are taken from any particular edge (i,j) to estimate its expected value?
   (b) How sample costs of an edge (i,j) from its given probability distribution q_{ij}?
   (c) What is the probability that the path found by the algorithm is the shortest?
2. The values of edge costs are never learned or become known after the shortest path is chosen. In this formulation, finding shortest path deals with finding a path that minimizes the expectation of a utility function that depends nonlinearly on the edge costs [6].
3. In the third formulation, edge costs are learned progressively as the graph is traversed. There are some difficulties with this approach such as
   (a) How many time the graph is traversed?
   (b) How many samples are taken from any edge?
   (c) How sample costs of an edge from its given probability distribution?
   (d) What is the probability that the path found by algorithm is the shortest?

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.

The analysis of some of the problems for general stochastic graphs has been attempted by Pritsker [7] and Frank [5]. Martin [8] has analyzed the problem using the unique arcs concept, and Sigal and et al.[9] have used the uniformly directed cuts in their analysis of shortest paths. Each one of the above four papers represents the required probabilistic quantities as multiple integrals. Numerical evaluation of these integrals quickly gets out of hand, even for small networks. Michandani [10] uses an entirely different approach that avoids the evaluation of multiple integrals, but his approach works only when the arc lengths are discrete random variables. Due to the computational difficulties in the exact computation of the distribution of the length of the shortest path, recently attention has shifted to Monte Carlo simulation of stochastic networks to obtain estimates of the required distributions. Simulation methods for estimating some of the quantities listed above are developed by Sigal and et al. [11] and Adlakha and Fishman [12, 13]. Two distributed learning automata (DLA) based algorithms for finding shortest path in stochastic graph are reported in the literature [1, 2]. In [1], 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 distributed learning automata is constructed from given input graph and each learning automaton in the DLA updates its action probability vector using L_{R-I} reinforcement scheme until the shortest path is found. In [2], another DLA based algorithm which is faster than the algorithm reported in [1] 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 [2] and the one reported in [1] is the definition of dynamic threshold. In [1], the dynamic threshold is the average cost of sampled paths whereas in the algorithm given in [2] 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 distributed learning automata (DLA) for solving the stochastic shortest path problem. We give an iterative stochastic algorithm that find the minimum expected value of set $\Pi = \{\pi_1, \ldots, \pi_r\}$ 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, distributed learning automata 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 [1, 2]) is used to prove that if each learning automaton used in distributed learning automata uses the L_{R-I} learning algorithm, the given algorithm finds the shortest path with probability as close as to the unity. The proof method for convergence reported in [1, 2] is

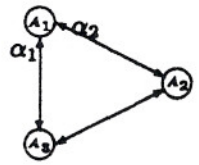

Figure 1: Distributed Learning Automata

based on Martingle theorem whereas the proof method for convergence 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 [1, 2]) for some of the graphs and perform worse for some other graphs.

The rest of the paper is organized as follows: The learning automata and distributed learning automata are introduced in section 2. Section 3 presents the proposed Algorithm. Simulation results and discussion are given in section 4 and section 5 concludes the paper.

2 Distributed Learning Automata

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. For example in figure 1, every automaton has two actions. If automaton A_1 selects action α_1 then automaton A_3 will be activated. The activated automaton A_3 chooses one of its actions which in turn activates one of the automata connected to A_3. At any time only one automaton in the network will be active. Formally, a DLA with n learning automata 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.

3 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 the DLA. The output of DLA is a sequence of actions that represent a particular path in the stochastic graph. The environment uses the length of this path to produce its response. This response, depending on whether it is *favorable* or *unfavorable*, causes the actions along this path be *rewarded* or *penalized*. In the proposed algorithm, at first a network of learning automata which is isomorphic to the input graph is created. In this network each node is a learning automaton and each outgoing edge of this node is one of the actions of this learning automaton. 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 3 to describe the algorithm. The action probability vector for learning automaton A_j is shown by $p^j = (p^j_1, p^j_2, \cdots, p^j_{r_j})$ where p^j_m 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 3 is represented by $\{\alpha_5, \alpha_6\}$. Variables *pathLen* and *pathCost* represent the length and the cost of the recently traversed path, respectively. The recently traversed path is stored in array *path* where $path[k]$ denotes the kth node in that path. 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. In this algorithm, the statement $node.getAdjNode()$ selects an action from A_{node} based on the action probability vector p^{node} and the statement $node \leftarrow nextNode$ activates automaton $A_{nextNode}$. Now, we give a general description of the proposed algorithm (figure 2).

In the first step, the source automaton A_s chooses one of its actions, say α_m. This action activates automaton A_m on the other end of the 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 *dynamic threshold* (described later) and depending on the result of comparison, the value of dynamic threshold is updated. Dynamic threshold

```
procedure DLA-SSPP
  Input: Graph G = (V, E), v_s, v_d, threshold
  Output: path , pathLen, pathCost
  begin
    Construct the DLA from graph G
    set k ← 1, T ← MAXCOST.
    set AvgCost(0) ← Average cost of some random paths.
    repeat
      set pathCost ← 0
      set pathLen ← 1
      set node ← path[1] ← v_s
      while node ≠ v_d and pathLen < n do
        nextNode ← node.getAdjNode ()
        pathLen ← pathLen + 1
        pathCost ← pathCost + edgeCost (node, nextNode)
        path [pathLen]← nextNode, node ← nextNode
      end while
      if path[pathLen]=v_d then
        AvgCost(k) ← ((k-1)×AvgCost(k-1)+pathCost)/k
S4:     if AvgCost(k) < T then
S1:       T ← AvgCost(k)
          for m ← pathLen-1 downto 1 do
            Reward action path[m+1] of automaton A_path[m]
            Update a_k according to equation (1)
          end for
        end if
        k ← k + 1
      end if
    until probability of selected path ≥ threshold
  end DLA-SSPP
```

Figure 2: The proposed algorithm

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 smaller than dynamic threshold T, dynamic threshold is set to $AvgCost$ and all the selected actions of automata along the traversed path are rewarded. If $AvgCost$ is grater than dynamic threshold T, dynamic threshold 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} learning 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.

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

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 [14]. 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 3.1 If $q(n)$ evolves according to the given algorithm, then the proposed algorithm converges to the optimal path with a probability as close to unity as desired for graphs with unique optimal path.

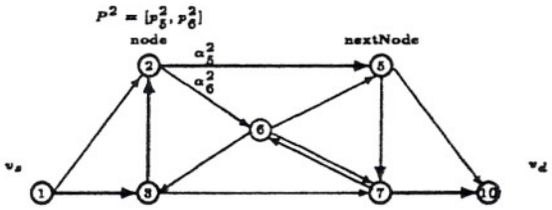

Figure 3:

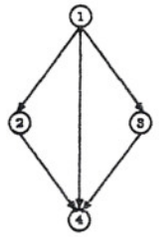

Figure 4: Graph 1

Proof The proof is given in [15].

4 Experiments

To study the feasibility of the proposed algorithms, experiments are conducted on different stochastic graphs given in figures 4 through 6. Graphs which are shown in figures 4 through 6 are borrowed from [16].

- Graph 1 which is shown in figure 4 is a graph with 4 nodes, 5 arcs, $v_s = 1$, and $v_d = 4$. Edge cost distribution is given in table 1.
- Graph 2 which is shown in figure 5 is a graph with 10 nodes, 23 arcs, $v_s = 1$, and $v_d = 10$. Edge cost distribution is given in table 2.
- Graph 3 which is shown in figure 6 is a graph with 15 nodes, 42 arcs, $v_s = 1$, and $v_d = 15$. Edge cost distribution is given in table 3.

To compare the performance of the proposed algorithm and the algorithm reported in [1, 2], three sets of experiments are conducted on three different stochastic graphs given in figures 4 through 6 and the results are summarized in tables 4 through 6. All three algorithms will be terminated when the probability of traversed path is greater than 0.9 or 900,000 paths are traversed. Each 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 [1, 2]) for some of the graphs and performs worse for some other graphs. No conclusion can be made about which of the algorithms perform the best all the time. It seems that the graph on which the algorithms are tested plays the major rule on the convergence of all three algorithms. For more experimentation refer to [15].

5 Conclusion

In this paper we have studied the problem of determining the optimal path in a stochastic network analogous to deterministic networks. The algorithms presented provide policy which can be used to determine a path from source to destination node with minimal expected cost. The algorithm presented

Table 1: Weight distribution of graph 1

Edge	Lengths				Probabilities			
(1,2)	6	14	15		0.3	0.45	0.25	
(1,3)	2	9	10		0.1	0.7	0.2	
(2,4)	4	11	18	19	0.25	0.1	0.3	0.35
(2,4)	6	13	15		0.15	0.3	0.55	

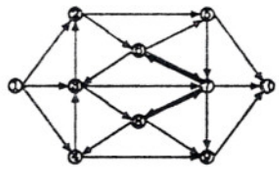

Figure 5: Graph 2

is adaptive procedure based on distributed learning automata. A new method of proof (different from [1, 2]) is used to prove that if each learning automaton used in distributed learning automata uses the L_{R-I} learning algorithm, the given algorithm finds the shortest path with probability as close as to the unity. The proof method reported in [1, 2] is based on Martingle theorem whereas the proof method used for the convergence of the proposed algorithm is based on sampling theory. The simulation results show that the proposed algorithm performs better in comparison with algorithms reported in [1, 2] for some of the graphs and performs worse for some other graphs. No conclusion can be made about which of the algorithms perform the best all the time. It seems the graph on which the algorithms are tested plays the major rule on the convergence of all three algorithms.

References

[1] M. R. Meybodi and H. Beigy, "Solving Stochastic Shortest Path Problem Using Distributed Learning Automata," in *Proceedings of 6th Annual International Computer Society of Iran Computer Conference CSICC-2001, Isfehan, Iran*, pp. 70–86, Feb. 2001.

[2] H. Beigy and M. R. Meybodi, "A New Distributed Learning Automata Based Algorithm For Solving Stochastic Shortest Path Problem," Tech. Rep. TR-CE-2001-006, Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran, 2001.

[3] N. Deo, , and C. Pang, "Shortest Path Algorithms: Taxonomy and anotation," *Newtorks*, vol. 14, pp. 275–323, 1984.

[4] G. H. Polychronopoulos and J. N. Tsitsiklis, "Stochastic Shortest Path Problems with Recourse," *Newtorks*, vol. 27, pp. 133–143, 1996.

[5] H. Frank, "Shortest Path in Probabilistic Graphs," *Operations Research*, vol. 17, pp. 583–599, 1969.

[6] P. B. Mirchandani and H. Soroush, "Shortest Distance and Reliability of Probabilistic Networks: A Case with Temporary Preferences," *Computer Operations Research*, vol. 12, no. 4, pp. 365–381, 1985.

[7] A. A. B. Pritsker, "Application of Multi-Channel Queuing Results to the Analysis of Conveyor Systems," *Journal of Industrial Engineering*, pp. 14–21, 1966.

[8] J. J. Martin, "Distribution of the Time through a Directed Acyclic Network," *Operations Research*, vol. 13, pp. 46–66, 1965.

[9] C. C. Sigal, A. A. B. Pritsker, and J. J. Solberg, "The Use of Cutsets in Monte-Carlo analysis of Stochastic Networks," *Math. Comp. Simulation*, vol. 21, pp. 376–384, 1979.

Table 2: Weight distribution of graph 2

Edge	Lengths				Probabilities			
(1,2)	3	5.3	7.4	9.4	0.2	0.2	0.3	0.2
(1,3)	3.5	6.2	7.9	8.5	0.3	0.3	0.2	0.2
(1,4)	4.2	6.1	6.9	8.9	0.2	0.3	0.2	0.3
(2,5)	2.6	4.1	5.5	9.0	0.2	0.2	0.4	0.2
(2,6)	5.8	7.0	8.5	9.6	0.3	0.3	0.2	0.2
(3,2)	1.5	2.3	3.6	4.5	0.2	0.2	0.3	0.3
(3,7)	6.5	7.2	8.3	9.4	0.5	0.2	0.2	0.1
(3,8)	5.9	7.8	8.6	9.9	0.4	0.3	0.1	0.2
(4,3)	2.1	3.2	4.5	6.8	0.2	0.2	0.3	0.3
(4,9)	1.1	2.2	3.5	4.3	0.2	0.3	0.4	0.1
(5,7)	3.2	4.8	6.7	8.2	0.2	0.2	0.3	0.3
(5,10)	6.3	7.8	8.4	9.1	0.2	0.2	0.4	0.2
(6,3)	6.8	7.7	8.5	9.6	0.4	0.1	0.1	0.4
(6,5)	0.6	1.5	3.9	5.8	0.2	0.2	0.3	0.3
(6,7)	2.1	4.8	6.6	7.5	0.2	0.4	0.2	0.2
(7,6)	4.1	6.3	8.5	9.7	0.2	0.3	0.4	0.1
(7,8)	1.6	2.8	5.2	6.0	0.2	0.3	0.3	0.2
(7,10)	1.6	3.4	8.2	9.3	0.2	0.3	0.3	0.2
(8,4)	7.0	8.0	8.8	9.4	0.2	0.2	0.2	0.4
(8,7)	2.1	4.6	8.5	9.6	0.4	0.2	0.2	0.2
(8,9)	1.7	4.9	6.5	7.8	0.2	0.2	0.2	0.4
(7,9)	3.5	4.0	5.0	7.7	0.1	0.2	0.4	0.3
(9,10)	4.6	6.4	7.6	8.9	0.4	0.1	0.2	0.3

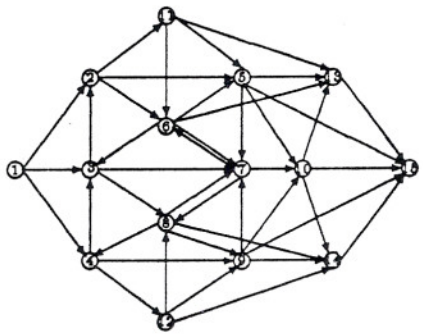

Figure 6: Graph 3

[10] P. B. Mirchandani, "Shortest Distance and Reliability of Probabilistic Networks," *Computer Operations Research*, vol. 3, pp. 347–355, 1976.

[11] C. C. Sigal, A. A. B. Pritsker, and J. J. Solberg, "The Stochastic Shortest Route Problem," *Operations Research*, vol. 28, pp. 1122–1129, 1980.

[12] G. S. Fishman, "Estimating Critical Parh and Arc Probabilities in Stochastic Activity Networks," Tech. Rep. TR-83/5, University of North Carolina, Chapel Hill, N.C. 27514, USA, 1983.

[13] V. G. Adlakha and G. S. Fishman, "An Improved conditional Monte Carlo Technique For The Stochastic Shortest Route Problem," Tech. Rep. TR-84-9, University of North Carolina, Chapel Hill, N.C. 27514, USA, 1984.

[14] K. Najim and A. S. Pozyak, *Learning Automata: Theory and Applications.* Oxford: Pergamon press, 1994.

[15] M. R. Meybodi and H. Beigy, "A Sampling Method Based on Distributed Learning Automata for Stochastic Shortest Path Problem," Tech. Rep. TR-CE-2001-007, Computer Engineering Department, Amirkabir University Technology, Tehran, Iran, 2001.

[16] C. Alexopoulos, "State Space Partitioning Methods for Stochastic Shortest Path Problems," *Newtorks*, vol. 30, pp. 9–21, 1997.

Table 3: Weight distribution of graph 3

Edge	Lengths				Probabilities			
(1,2)	16	25	36		0.6	0.3	0.1	
(1,3)	21	24	25	39	0.5	0.2	0.2	0.1
(1,4)	11	13	26		0.4	0.4	0.2	
(2,11)	24	28	31		0.5	0.3	0.2	
(2,5)	11	30			0.7	0.3		
(2,6)	13	37	39		0.6	0.2	0.2	
(3,2)	11	20	24		0.6	0.3	0.1	
(3,7)	23	30	34		0.4	0.3	0.3	
(3,8)	14	23	34		0.5	0.4	0.1	
(4,3)	22	30			0.7	0.3		
(4,9)	35	40			0.6	0.4		
(4,12)	16	25	37		0.5	0.4	0.1	
(5,13)	28	35	37	40	0.4	0.3	0.2	0.1
(5,15)	25	32			0.7	0.3		
(5,10)	27	33	40		0.4	0.3	0.3	
(5,7)	15	17	19	26	0.3	0.3	0.3	0.1
(6,5)	18	25	29		0.5	0.3	0.2	
(6,13)	21	23			0.5	0.5		
(6,7)	11	31	37		0.5	0.5	0.1	
(6,3)	18	24			0.7	0.3		
(7,10)	19	23	37		0.6	0.2	0.2	
(7,8)	12	15	22	24	0.3	0.3	0.3	0.2
(7,6)	12	23	31		0.5	0.3	0.2	
(8,7)	14	34	39		0.6	0.2	0.2	
(8,14)	14	15	27	32	0.3	0.3	0.2	0.2
(8,9)	13	31	32		0.8	0.1	0.1	
(8,4)	13	23	34		0.4	0.3	0.3	
(9,7)	10	17	20		0.6	0.3	0.1	
(9,10)	16	18	36	39	0.3	0.3	0.2	0.2
(9,15)	12	13	25	32	0.4	0.3	0.2	0.1
(9,14)	19	24	29		0.4	0.3	0.3	
(10,13)	14	20	25	32	0.3	0.3	0.2	0.2
(10,15)	15	19	25		0.4	0.3	0.3	
(10,14)	23	34			0.9	0.1		
(11,13)	13	31	25		0.6	0.3	0.1	
(11,5)	18	19	20	23	0.3	0.3	0.3	0.1
(11,6)	10	19	39		0.5	0.4	0.1	
(12,8)	15	36	39		0.5	0.3	0.2	
(12,9)	16	22			0.7	0.3		
(12,14)	10	13	18	34	0.3	0.3	0.3	0.1
(13,15)	12	31			0.9	0.1		
(14,15)	14	19	32		0.5	0.3	0.2	

Table 4: The simulation results of Algorithm($\epsilon = 0.01$)

a	Graph 1		Graph 2		Graph 3	
	Average Iterations	Runs Converged	Average Iterations	Runs Converged	Average 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

a	Graph 1		Graph 2		Graph 3	
	Average Samples	Shortest Path	Average Samples	Shortest Path	Average 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

Table 5: The simulation results of algorithm reported in [1]

a	Graph 1		Graph 2		Graph 3	
	Average Iterations	Runs Converged	Average Iterations	Runs Converged	Average Iterations	Runs Converged
0.0060	758	100	8547	100	13133	100
0.0061	739	100	8694	100	13518	100
0.0062	728	100	8376	100	12905	100
0.0062	717	100	8621	99	12502	100
0.0063	705	100	7997	100	12789	100
0.0064	693	100	7785	100	12901	100
0.0065	688	100	7904	100	13195	100
0.0066	679	100	7616	100	11100	100
0.0068	663	100	7579	99	11101	100
0.0069	656	100	7376	100	12274	100
0.0070	649	100	7634	100	11325	100

a	Graph 1		Graph 2		Graph 3	
	Average Samples	Shortest Path	Average Samples	Shortest Path	Average Samples	Shortest Path
0.0060	1521	1093	27742	13090	44473	26250
0.0061	1482	1070	28136	13161	45617	26952
0.0062	1466	1053	27159	12831	43643	25259
0.0062	1439	1038	27920	12807	42366	24512
0.0063	1415	1022	25936	12079	43138	25819
0.0064	1391	1004	25284	11993	43396	25785
0.0065	1381	990	25632	11960	44251	25896
0.0066	1352	977	24712	11668	37822	22251
0.0068	1330	961	24715	11649	37777	21903
0.0069	1321	949	24949	11381	41248	24191
0.0070	1303	935	24716	11391	38317	22665

Table 6: The simulation results of algorithm reported in [2]

a	Graph 1		Graph 2		Graph 3	
	Average Iterations	Runs Converged	Average Iterations	Runs Converged	Average Iterations	Runs Converged
0.0060	623	100	2188	91	3303	69
0.0061	623	100	2254	95	3606	70
0.0062	605	100	1950	98	3488	72
0.0063	599	100	2289	96	3419	74
0.0064	597	100	2002	94	3247	68
0.0065	606	100	1892	97	3371	68
0.0066	571	100	2232	93	3194	69
0.0067	591	100	1882	94	2852	67
0.0068	569	100	1920	90	3271	72
0.0069	563	100	2145	92	3266	75
0.0070	563	100	1933	90	3003	74

a	Graph 1		Graph 2		Graph 3	
	Average Samples	Shortest Path	Average Samples	Shortest Path	Average Samples	Shortest Path
0.0060	1244	785	7491	2972	16372	4701
0.0061	1244	781	7362	2960	17946	5344
0.0062	1208	764	6266	2784	15467	5424
0.0063	1196	752	7260	2990	16396	5406
0.0064	1192	749	6576	2796	17220	4519
0.0065	1212	743	6079	2743	14766	5003
0.0066	1140	719	7216	2903	13664	4372
0.0067	1180	729	6292	2671	13818	4322
0.0068	1136	712	6517	2626	14642	4774
0.0069	1126	702	7078	2764	14289	4959
0.0070	1126	697	6401	2613	12528	4179

THE 10TH IRANIAN CONFERENCE ON ELECTRICAL ENGINEERING

MAY 14-16, 2002

PROCEEDINGS

ELECTRONICS
COMPUTER
BIOMEDICAL

VOLUME 1

DEPARTMENT OF ELECTRICAL ENGINEERING
FACULTY OF ENGINEERING
UNIVERSITY OF TABRIZ, TABRIZ, IRAN