International Journal of

UNCERTAINTY, FUZZINESS AND KNOWLEDGE-BASED SYSTEMS

Volume 14 • Number 5 • October 2006

Utilizing Distributed Learning Automata to Solve Stochastic Shortest Path Problems

H. Beigy & M. R. Meybodi

Online version at:
http://www.worldscinet.com/ijufks.html

World Scientific

NEW JERSEY • LONDON • SINGAPORE • SHANGHAI • HONG KONG • TAIPEI • CHENNAI

International Journal of Uncertainty,
Fuzziness and Knowledge-Based Systems
Vol. 14, No. 5 (2006) 591–615
© World Scientific Publishing Company

World Scientific
www.worldscientific.com

UTILIZING DISTRIBUTED LEARNING AUTOMATA TO SOLVE STOCHASTIC SHORTEST PATH PROBLEMS

HAMID BEIGY*
Computer Engineering Department, Sharif University of Technology, Tehran, Iran
Institute for Studies in Theoretical Physics and Mathematics (IPM),
School of Computer Science, Tehran, Iran
beigy@ce.sharif.edu

M. R. MEYBODI
Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran
Institute for Studies in Theoretical Physics and Mathematics (IPM),
School of Computer Science, Tehran, Iran
mmeybodi@aut.ac.ir

Received 10 June 2003
Revised 22 July 2006

In this paper, we first introduce a network of learning automata, which we call it as distributed learning automata and then propose some iterative algorithms for solving stochastic shortest path problem. These algorithms use distributed learning automata to find a policy that determines a path from a source node to a destination node with minimal expected cost (length). In these algorithms, at each stage distributed 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. It is shown that the shortest path is found with a probability as close as to unity by proper choice of the parameters of the proposed algorithms.

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

1. Introduction

Shortest path problems when the edge lengths are deterministic can be solved using one of the many shortest path algorithms, such as Dijkstra and Floyd-Warshall in polynomial time [1]. However, when the edge lengths are allowed to be random, the problem becomes considerably more difficult. Unlike the case for deterministic graphs, stochastic graphs are related to distinct models, interpretations and applications. Recently, attention has been given to the analysis of shortest paths in such stochastic networks, and the following questions have been addressed.

*This research was in part supported by a grant from Institute for Studies in Theoretical Physics and Mathematics (IPM), Tehran, Iran.

(1) distribution of the length of the shortest path
(2) moments of the length of the shortest path
(3) the probability that a given path is the shortest path in the graph

Stochastic shortest path problems can be grouped in two main classes: the first class aims to find an priori solution that minimizes the expected lengths, while the second one computes an online solution that allows decisions to be made at various stages (also known as a recourse problem). Much of researches on first class can be reviewed as follows. The analysis of some of the problems for general stochastic graphs has been attempted by Pritsker [2] and Frank [3]. Martin [4] has analyzed the problem using the unique arcs concept, and Sigal et al.[5] 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. Mirchandani [6] 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 et al. [7] and Adlakha and Fishman [8,9]. In [10], state space partitioning was used for solving the stochastic shortest path problem. Polychronopoulos and Tsitsiklis proposed two models with recourse [11]. The first model assumes general spatial dependence among the edges captured in the form of realizations, representing all possible network instantiations. An exponential algorithm with respect to the number of realizations is proposed in this model. The second model is a special case of the first model where the edge lengths are independently distributed random variables. An exponential algorithm with respect to the number of edges was given for this model. Two algorithms for the stochastic shortest path problem with recourse on time varying networks are proposed in [12,13]. In [14], the stochastic shortest path problem with recourse is considered, where it is a limited inter-edge dependencies. The solution to this problem consists of a collection of state-paths, each associated with a cost and probability for being followed. It must be noted that in all these studies, it is assumed that the cost distributions for edges in the graph are known in advance.

In this paper an adaptive procedure based on the distributed learning automata (DLA) for finding the shortest path in a stochastic graph is presented. This problem is particularly useful in routing in transportation or communication networks and in scheduling. For example in transportation applications the random attribute denotes a random travel time.

In this paper, DLA is introduced and applied to the shortest path problem. In order to compute the probability that a path being the shortest, a distributed learning automaton is constructed from the 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. It has been shown that one of the proposed algorithm finds the shortest path in a stochastic graph with a probability as close as to unity by proper choice of the parameters of the proposed algorithms. The simulation results also confirm the theoretical results.

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

2. Stochastic Graph, Learning Automata, and Distributed Learning Automata

2.1. *Stochastic Graph*

A stochastic graph G is defined by a triple $G = (V, E, \mathcal{F})$, where $V = \{1, 2, \ldots, n\}$ is set of nodes, $E \subset V \times V$ specifies set of edges, and $n \times n$ matrix $\mathcal{F}$ is the probability distribution describing the statistics of edge lengths, where n is the number of nodes. In particular, length C_{ij} of edge (i, j) is assumed to be a positive random variable with f_{ij} as its probability density function, which is assumed to be unknown for the proposed algorithms. In stochastic graph G, a path π_i with length of n_i nodes and expected length of $\bar{L}_{\pi_i}$ from source node v_s to destination node v_d is defined as an ordering $\{\pi_{i,1}, \pi_{i,2}, \ldots, \pi_{i,n_i}\} \subset V$ of nodes in such a way that $v_{\pi_{i,1}} = v_s$ and $v_{\pi_{i,n_i}} = v_d$ are source and destination nodes, respectively and $(\pi_{i,j}, \pi_{i,j+1}) \in E$ for $1 \leq j < n_i$, where $\pi_{i,j}$ is the j^{th} node in path π_i. Assume that there are r distinct paths $\Pi = \{\pi_1, \pi_2, \ldots, \pi_r\}$ between v_s and v_d. The shortest path between source node v_s and destination node v_d denoted by $v_s \sim v_d$, is defined as a path with minimum expected length. In other word, the shortest path π^* has length of $\bar{L}_{\pi^*} = \min_{\pi \in \Pi}\{\bar{L}_\pi\}$.

2.2. *Learning Automata*

The automata approach to learning involves the determination of an optimal action from a set of allowable actions. Automaton selects an action from its finite set of actions, which serves as the input to the environment which in turn emits a stochastic response $\beta(k) \in \{0, 1\}$ at the time k. The environment penalizes (i.e. $\beta(k) = 1$) action α_i of the automaton with the penalty probability $c_i (d_i = 1 - c_i)$. On the basis of $\beta(k)$, the state of the automaton is updated and a new action chosen at $(k + 1)$. Note that the c_i are unknown initially and it is desired that as a result of the interaction between the automaton and the environment the automaton arrives at the action which presents it with the minimum penalty response in an expected sense. An automaton acting in an unknown random environment and improves its performance in some specified manner, is referred to as a *learning automaton* (LA). 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 a set of inputs, $\underline{\alpha}$ is a set of actions, and T is learning algorithm. The learning algorithm is a recurrence relation and is used to modify the state probability vector. Let α_i be the action chosen at time k as a sample realization from the probability distribution $\underline{p}(k)$. The linear reward-inaction algorithm (L_{R-I}) is one of the earliest learning schemes reported in the literature and its recurrence equation for updating the action probability vector $\underline{p}$ is defined as

$$p_j(k+1) = \begin{cases} p_j(k) + a(k) \times [1 - p_j(k)] & \text{if} \quad i = j \\ p_j(k) - a(k) \times p_j(k) & \text{if} \quad i \neq j \end{cases} \tag{1}$$

if $\beta(k)$ is zero and $\underline{p}$ is unchanged if $\beta(k)$ is one. $a(k) \in (0,1)$ is *step length* at the stage k and determines the amount of increases (decreases) of the action probabilities. In linear reward-ϵpenalty ($L_{R-\epsilon P}$) scheme the recurrence equation for updating $\underline{p}$ is defined as

$$p_j(k+1) = \begin{cases} p_j(k) + a(k) \times [1 - p_j(k)] & \text{if} \quad i = j \\ \\ p_j(k) - a(k) \times p_j(k) & \text{if} \quad i \neq j \end{cases} \tag{2}$$

when $\beta(k)$ is zero and

$$p_j(k+1) = \begin{cases} p_j(k) \times (1 - b(k)) & \text{if} \quad i = j \\ \\ \frac{b}{r-1} + p_j(k)(1 - b(k)) & \text{if} \quad i \neq j \end{cases} \tag{3}$$

when $\beta(k)$ is one. $0 < b(k) \ll a(k)$ represents *step length* at the stage k and r is the number of actions for learning automata. $a(k)$ and $b(k)$ determine the amount of increase and decreases of the action probabilities, respectively. If the $a(k)$ equals to $b(k)$, then recurrence equations (2) and (3) is called *linear reward penalty*(L_{R-P}) algorithm.

A variety of learning algorithms such as discretized, pursuit, and estimator learning algorithms are proposed in [15,16,17]. Learning automaton have been used successfully in many applications such as telephone and data network routing [18,19], solving NP-Complete problems [20,21,22], capacity assignment [23] and neural network engineering [24,25,26,27] to mention a few. For more information about learning automata refer to [28].

2.3. *Distributed Learning Automata*

In this section, we introduce a new model of interconnected automata, which we call it distributed learning automata (DLA). Distributed learning automata is a network of automata which collectively cooperate to solve a particular problem. A DLA can be modeled by a directed graph in which the set of nodes of graph constitute the set of automata and the set of outgoing edges for each node constitutes the set of actions for corresponding 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. An example of DLA is given in figure 1, in which every automaton has two actions. If automaton A_1 selects action α_3, then automaton A_3 will be activated. 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 activated. 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. Let action probability vector for learning automaton A_j is represented by $\underline{p}^j$ where a component p^j_m of $\underline{p}^j$ denotes the probability of choosing action α_m, that is, the probability of choosing edge (j, m).

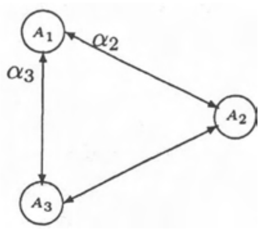

Fig. 1. Distributed Learning Automata

3. The Proposed Algorithm

In this section, we propose several distributed learning automata based algorithms for finding a path with minimum expected length (shortest path) in a stochastic graph. In these algorithms, the stochastic graph plays the role of random environment for the DLA. The action of a DLA is a sequence of actions that represents a particular path in the stochastic graph. The environment uses the sampled length of this path to produce its response. This response, depending on whether it is *favorable* or *unfavorable*, causes the actions along the traversed path be *rewarded* or *penalized.*

The first algorithm is given in algorithm 1 and can be described as follows. 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. Then, at the stage k, source automaton (corresponding to the source node in the graph) A_s chooses one of its actions (as a sample realization of its action probability vector), say action α_m. This action activates automaton A_m on the other end of edge (s, m). The process of choosing 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 A_d is reached, length of the traversed path, (L_{π_i}), is computed and then compared with a quantity called *dynamic threshold*, T_k. Depending on the result of the comparison all the learning automata (except the destination learning automaton) along the traversed path update their action probabilities. Updating is done in direction from source to destination or vice versa. If length of the traversed path is less than or equal to the dynamic threshold then all learning automata along that path receive reward and if length of the traversed path is greater than the dynamic threshold or the destination node is not reached, then activated automata receive penalty.

Algorithm 1 The first algorithm for solving stochastic shortest path problem.

```
Input: Stochastic graph $G = (V, E, \mathcal{F})$, start node $v_s$, destination node $v_d$, thresholds
  $P_{pop}$ and $K$
Output: The shortest path
  Construct a DLA from graph $G$
  Let $k$ be the stage number and initially set to 0
  Let $T_k$ be the dynamic threshold at stage $k$ and initially set to 0
  repeat
    Let $u$ be source node $v_s$
    while ($v_d$ is not reached and the number of traversed edges is less than $|V|$) do
      Select an adjacent node to $u$ according to the action probability vector of $A_u$ and
      denote it $w$
      Add cost of edge $(u, w)$ to the cost of traversed path and set $u$ to $w$
    end while
    if ($v_d$ is reached) then
      Let $\pi_i$ and $L_{\pi_i}$ be the traversed path and its cost, respectively
      if ($L_{\pi_i}$ is less than $T_k$ ) then
        Reward the selected actions of activated automata along path $\pi_i$
      else
        Penalize the selected actions of activated automata along path $\pi_i$
      end if
      Increment stage number k
      Compute $T_k$ as average cost of traversed paths from $v_s$ to $v_d$ (i.e. $T_{k-1} +
      \frac{1}{k}[L_{\pi_i} - T_{k-1}]$)
    end if
  until (probability of path $\pi_i$ is greater than a pre-specified threshold $P_{pop}$ or $k$ is greater
  than $K$ )
```

The process of traveling from the source learning automaton to the destination learning automaton is repeated until the stopping condition is reached which at this point the last traversed path is the path which has the minimum expected length among all paths from the source to the destination. Assume that up to the stage k, path π_i (for $i = 1, 2, \cdots, r$) is traversed k_i times, where $k = \sum_{i=1}^{r} k_i$. Also assume that path π_i is selected at the stage k. Let $L_{\pi_i}(j)$ and $\bar{L}_{\pi_i}$ be the length of the path

π_i (for $i = 1, 2, \cdots, r$) at the j^{th} sampling time (for $j = 1, 2, \cdots, k_i$)and the mean of the length of the path π_i ($i = 1, 2, \cdots, r$), respectively. Thus the dynamic threshold at instant k can be defined by equation 6. According to algorithm 4, we have

$$c_i(k) = \text{Prob}\ [L_{\pi_i}(k+1) > T(k)] \tag{4}$$

and

$$d_i(k) = \text{Prob}\ [L_{\pi_i}(k+1) \leq T(k)], \tag{5}$$

where $c_i(k)$ and $d_i(k)$ denote penalty and reward probabilities of path π at the stage k, respectively. The dynamic threshold at time $k > 1$ is defined as

$$T_k = \frac{1}{n} \sum_{i=1}^{r} T_i(k_i), \tag{6}$$

where k_i s the number of times that path π_i is traversed and

$$T_i(k_i) = \frac{1}{k_i} \sum_{j=1}^{k_i} L_{\pi_i}(j).$$

The algorithm stops if the product of the probability of choosing the edges of the traversed path, called *path probability*, is greater than a certain threshold or a pre-specified number of paths are traversed.

For the sake of simplicity, we use figure 2 for discussing the first algorithm. In the stage k, A_1 chooses one of its actions, say α_3, which activates A_3. A_3 using its action probability vector will activate A_2. The process of selection of an action and activating an automaton is repeated until the A_{10} 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, i.e. 10. Assume that in figure 2, the path with tick lines is traversed, that is, $\pi_i = (1, 3, 2, 5, 7, 10)$. After the destination node is reached, the length of the traversed path is computed and then compared with the dynamic threshold and depending on the result of comparison all the learning automata (except A_{10}) along the traversed path update their action probabilities. If length of the traversed path is less than or equal to the dynamic threshold then all the activated automata along the traversed path receive reward and otherwise receive penalty. In the rest of this paper, we call this algorithm, algorithm 1.

This algorithm can be improved in several ways some of which are described below.

- In algorithm 1, the number of actions for every automaton is fixed and does not vary as the algorithm proceeds. This may cause that the algorithm to traverse paths that contain loops. In order to solve this problem, we use learning automata with variable number of action [29] by disabling actions corresponding to the visited nodes if any. To implement this, when an automaton chooses action k from the list of its actions, all the inactivated learning automata will disable

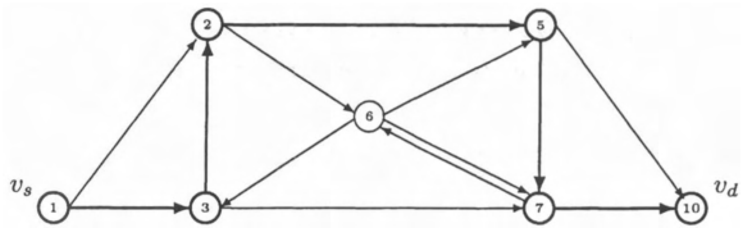

Fig. 2. An example graph

action k (but not removed) in their list of actions. When action probability vectors of automata in the traversed path is being updated, all the disabled actions will be enabled. Algorithm 1 in which this modification is made is called algorithm 2.

- In algorithms 1 and 2, learning automata use the same learning rate which remains fixed during the execution of the algorithm. This gives equal importance to all nodes along the path being traversed, which may not be appropriate. Nodes which are closer to the source node should be given higher degree of participation in the optimal path (higher probability of being part of the optimal path) than those which are far from the source node. Therefore, the amount of the reward given to an action taken at a particular time must increase as we approaching the destination node. To achieve this, we halve the step length as moving from one node to another when traversing back toward the source node. Algorithm 2 in which this modification is made will be called algorithm 3.
- Updating schemes for the step lengths can be different. Another way to update the step lengths is given by equation (7). Here the step lengths are dependent on the action probabilities. Algorithm 2 in which this modification is made will be called algorithm 4. In this algorithm, updating is done from the source node to the destination node.

$$a_{\pi_{i,j}}(k) = \frac{a_{\pi_{i,j-1}}(k)}{p_{\pi_{i,j}}^{\pi_{i,j-1}}(k+1)}, \tag{7}$$

where $a_{\pi_{i,j}}(k)$ denotes the step length of j^{th} automaton along path π_i at the k^{th} stage (i.e. $A_{\pi_{i,j}}$). It can be shown that $0 < a_{\pi_{i,j}}(k) < 1$. Similar approach for updating step length a in hierarchical system of learning automata has been reported in [30].
- In the previous algorithms, the length of the traversed path at any stage is computed using the length of the edges sampled at that stage. The algorithm for computation of the length of the traversed path at any stage may use the mean of the sampled length of an edge up to that stage. That is, the length of an edge in this version of the algorithm at the stage k is the average of sampled edge lengths up to the stage k. This modification, according to the central limit

theorem [31], allows the algorithm to have an estimate of the length of traversed path which is closer to its mean value. Algorithm 2 in which this modification is made will be called algorithm 5.

- Different combination of above mentioned algorithms can be used. For example, we examine another algorithm called algorithm 6 which is a combination of algorithms 3 and 5.

Theorem 1 shows the convergence for algorithm 4 when L_{R-I} learning automaton is used in each node. Our method of proving theorem 1, is very similar to the method given in [30,32,33] in the context of the analysis of learning automata operating in non-stationary environments. Before we state and prove the main result, we prove two lemmas.

Lemma 1. *Let $q_i(k)$ be the probability of traveling along path π_i at the stage k in a stochastic graph. If $\underline{q}(k)$ evolves according to the proposed algorithm, then for every $\delta > 0$ and $M > 0$, there exists a $a^* \in (0,1)$ and $K_0 < \infty$ such that for all $a \in (0,a^*)$* Prob [all paths are traversed at least M times before stage k] $> 1 - \delta$, *for all $k > K_0$.*

Outline of the Proof: Let random variable X_k^i be the number of times path π_i is traversed in any specific realization. If $\sum_k q_i(k)$ almost surely diverges then the lemma trivially follows for path π_i. Hence assume that the set of realizations over which $\sum_k q_i(k)$ converges is not a null set. Now for each realization belonging to this set it can be shown that $\mathrm{E}\left[X_k^i\right]$ can be made arbitrarily large values for sufficiently large k by taking a sufficiently small. Also, with k, $\mathrm{E}\left[X_k^i\right]$ increases faster than does $\mathrm{Var}\left[X_k^i\right]$. Now the lemma follows from Chebeshev inequality.

Lemma 2. *Assume that path π_i is traversed at the stage k. Let $c_i(k) = \mathrm{Prob}[L_{\pi_i} > T_k]$ and $c_i^* = \lim_{k\to\infty} c_i(k)$ (for $i = 1, 2, \ldots, r$) and let $\bar{m}_i(k)$ be the number of times path π_i is traversed up to the stage k. Then for any $\epsilon, \delta \in (0,1)$, we have*

$$\mathrm{Prob}\left[|c_i^* - c_i(k)| > \epsilon\right] < \delta,$$

for all $k > K(\epsilon,\delta)$, $a < a^*(\epsilon,\delta)$, and $i = 1, 2, \ldots, r$.

Outline of the Proof: Let Δ be the difference between two smallest elements of $\{c_1^*, \ldots, c_r^*\}$. Since the graph has a unique shortest path, we have $\Delta > 0$. Using weak law of large numbers, we conclude that there exists a $k_i(\Delta,\delta) < \infty$ such that

$$\mathrm{Prob}\left[|c_i^* - c_i(k)| < \frac{\Delta}{2}\right] > 1 - \delta,$$

for all k such that $\bar{m}_i(k) > k_i(\Delta,\delta)$. By definition of Δ, we have

$$\mathrm{Prob}\left[c_j(k) > c_i(k)\right] > 1 - \delta,$$

for all $j \neq i$ and for all k such that $\min_l\{\bar{m}_l(k)\} > M$, where $M = \max_l\{k_l(\Delta,\delta)\}$. Using lemma 1, we can find K_0 and a^* such that

$$\mathrm{Prob}\left[\min_i\{\bar{m}_i(k)\} > M\right] > 1 - \delta,$$

for all $k > K_0$ and all $a \in (0, a^*)$, and hence the proof of the lemma.

Theorem 1. *Let $q_i(k)$ be the probability of travelling along path π_i at the stage k in a stochastic graph. If $\underline{q}(k)$ evolves according to algorithm 4, then for every $\epsilon > 0$, there exists a step length $a^* \in (0,1)$ such that for all $a \in (0, a^*)$, we have*

$$Prob[lim_{k\to\infty} q_i(k) = 1] \geq 1 - \epsilon.$$

Stated in other words, the above theorem asserts that the proposed algorithm converges to the shortest path with a probability as close to unity as desired.

Outline of the Proof: This theorem is proved in the following three stages: 1) $q_i(k)$ (for $i = 1, \ldots, r$) are computed in terms of action probabilities of learning automata in path π_i, 2) it is shown that the probability of choosing the shortest path is a sub-Martingale process for large enough k, and 3) the convergence is shown using Martingale convergence theorems.

The complete details of the proof of theorem 1 depends on the specific graph. In order to show these details, we prove the theorem for graph given in figure 3 with $v_s = 1$, $v_d = 5$, and shortest path $\pi^* = (1, 2, 3, 4, 5)$. Before we present the proof, we prove some lemmas and a corollary.

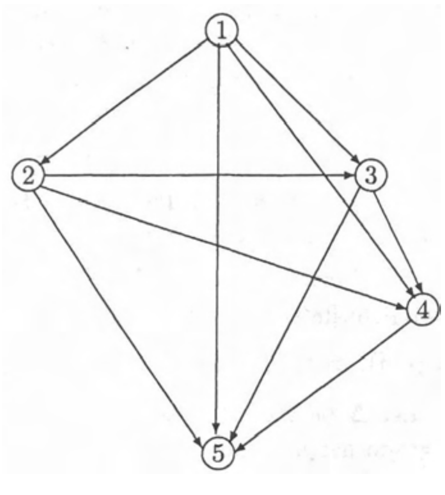

Fig. 3. Sample graph

Lemma 3. *If $\underline{q}(k)$ evolves according to algorithm 4 with L_{R-I} learning algorithm, then we have*

$$E[q_4(k+1) - q_4(k) | q(k)] = a q_4(k) \sum_{j \neq 4} q_j(k)[c_j(k) - c_4(k)]$$
$$+ a_3(k)(1-a)[q_3(k) q_8(k) d_8(k) - q_4(k) q_7(k) d_7(k)],$$

where $q_4(k)$ is the probability of travelling along path π_4.

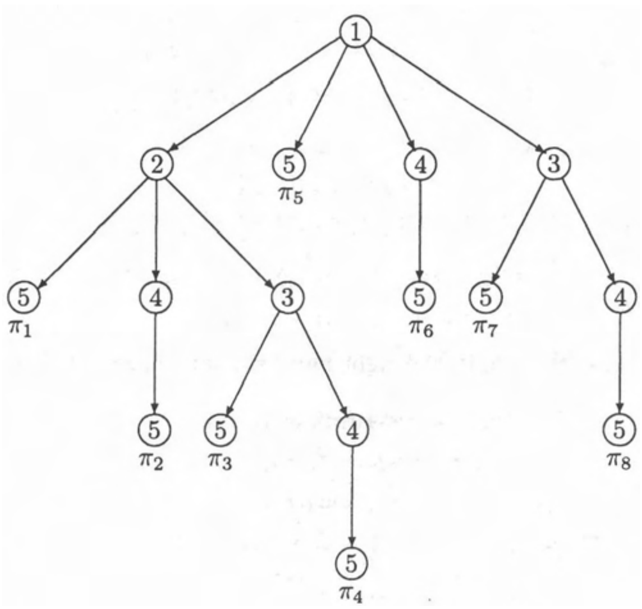

Fig. 4. Search tree for graph of figure 3

Proof: From the proposed algorithm, the search tree for finding path $v_s \sim v_d$ of graph given in figure 3 is shown in figure 3. Let $q_j(k)$ be the probability of traversing path π_j at the stage k. Therefore we have

$$\begin{aligned}
q_1(k) &= p_2^1(k)p_5^2(k)\\
q_2(k) &= p_2^1(k)p_4^2(k)p_5^4(k)\\
q_3(k) &= p_2^1(k)p_3^2(k)p_5^3(k)\\
q_4(k) &= p_2^1(k)p_3^2(k)p_4^3(k)p_5^4(k)\\
q_5(k) &= p_5^1(k)\\
q_6(k) &= p_4^1(k)p_5^4(k)\\
q_7(k) &= p_3^1(k)p_5^3(k)\\
q_8(k) &= p_3^1(k)p_4^3(k)p_5^4(k),
\end{aligned} \tag{8}$$

where $p_j^i(k)$ denotes the probability of selecting action α_j of automaton A_i at the stage k. Define $\Delta q_4(k) = E[q_4(k+1) - q_4(k)|q(k)]$. Therefore we obtain equation (9), where $a_i(k)$ is the step length of automaton A_i at the stage k and is computed by equation (7).

$$\begin{aligned}\Delta q_4(k) = {} & q_1(k)\left[c_1(k)q_4(k) + d_1(k)\{p_2^1(k) + a(1-p_2^1(k))\}p_3^2(k)(1-a_2(k))p_4^3(k)\right] \\ & + q_2(k)\left[c_2(k)q_4(k) + d_2(k)\{p_2^1(k) + a(1-p_2^1(k))\}p_3^2(k)(1-a_2(k))p_4^3(k)\right] \\ & + q_3(k)[c_3(k)q_4(k) + d_3(k) \\ & \times \{p_2^1(k) + a(1-p_2^1(k))\}\{p_3^2(k) + a_2(k)(1-p_3^2(k))\}p_4^3(k)(1-a_3(k))] \\ & + q_4(k)[c_4(k)q_4(k) + d_4(k)\{p_2^1(k) + a(1-p_2^1(k))\} \\ & \times \{p_3^2(k) + a_2(k)(1-p_3^2(k))\}\{p_4^3(k) + a_3(k)(1-p_4^3(k))\}] \\ & + q_5(k)\left[c_5(k)q_4(k) + d_5(k)p_2^1(k)p_3^2(k)p_4^3(k)(1-a)\right] \\ & + q_6(k)\left[c_6(k)q_4(k) + d_6(k)p_2^1(k)p_3^2(k)p_4^3(k)(1-a)\right] \\ & + q_7(k)\left[c_7(k)q_4(k) + d_7(k)p_2^1(k)p_3^2(k)p_2^3(k)(1-a)(1-a_3(k))\right] \\ & + q_8(k)\left[c_8(k)q_4(k) + d_8(k)p_2^1(k)p_2^3(k)(1-a)\{p_4^3(k) + a_3(k)(1-p_4^3(k))\}\right] \\ & - q_4(k)\end{aligned} \tag{9}$$

After simplifying each term in the right hand side of equation (9), we obtain

$$\begin{aligned}\Delta q_4(k) = {} & q_1(k)\left[q_4(k) - aq_4(k)d_1(k)\right] \\ & + q_2(k)\left[q_4(k) - aq_4(k)d_2(k)\right] \\ & + q_3(k)\left[q_4(k) - aq_4(k)d_3(k)\right] \\ & + q_4(k)\left[q_4(k) + a(1-q_4(k))d_4(k)\right] \\ & + q_5(k)\left[q_4(k) - aq_4(k)d_5(k)\right] \\ & + q_6(k)\left[q_4(k) - aq_4(k)d_6(k)\right] \\ & + q_7(k)\left[q_4(k) - aq_4(k)d_7(k) - q_4(k)d_7(k)a_3(k)(1-a)\right] \\ & + q_8(k)\left[q_4(k) - aq_4(k)d_8(k) + q_3(k)d_8(k)a_3(k)(1-a)\right] \\ & - q_4(k).\end{aligned} \tag{10}$$

After some algebraic, we obtain

$$\begin{aligned}\Delta q_4(k) = {} & aq_4(k)\sum_{j\neq 4} q_j(k)[c_j(k) - c_4(k)] \\ & + a_3(k)(1-a)[q_3(k)q_8(k)d_8(k) - q_4(k)q_7(k)d_7(k)],\end{aligned} \tag{11}$$

which completes the proof of this lemma.

Lemma 4. *For the graph of figure 3, when π_4 is the path with minimum expected length and $\underline{q}(k)$ evolves according to algorithm 4, the increment in the conditional expectation of $q_4(k)$ is always non-negative, that is, $\Delta q_4(k) \geq 0$ for all $q_4(k) \in (0,1)$ ($\Delta q_4(k) > 0$ for all $q_4(k) \in S_8^o$) with equality holding only when $q_4(k) \in \{0,1\}$, where S_8^o denotes the 8-dimensional open simplex.*

Proof: Since π_4 is the shortest path, then $c_i^* - c_4^* > 0$ and from lemma 2, we have $c_i(k) - c_4(k) > 0$ for all $i \neq 4$ and for large values of k. Hence, the first term of equation (11) is positive for large values of k. It is evident from equation (11) if $a_3(k)(1-a)[q_3(k)q_8(k)d_8(k) - q_4(k)q_7(k)d_7(k)] \geq 0$, then $\Delta q_4(k)$ is positive.

Substituting $q_3(k)$, $q_4(k)$, $q_7(k)$, and $q_8(k)$ from equation (8) into above inequality, we have $c_7^* > c_8^*$. Since path π_4 is the shortest path in graph of figure 3, we have $c_7^* > c_8^*$. Because in this graph, when π_4 is the shortest path, then the average length of edge $(3, 5)$ is greater than the sum of average lengths of edges $(3, 4)$ and $(4, 5)$. Since edge $(1, 3)$ is part of both paths π_7 and π_8, we conclude that path π_8 is shorter than path π_7, which implies $c_7^* > c_8^*$ or $d_7^* <= d_7^*$. Hence, two terms in the right hand side of equation (11) is non-negative.

Corollary 1. $\lim_{k\to\infty} q_4(k) = q_4^*$ *exists and* $q_4^* \in \{0, 1\}$ *with probability one.*

Proof: Lemma 4 implies that $\{q(k)\}$ is a sub–Martingale. Using Martingale theorems, and the fact that $\{q(k)\}$ is non–negative and uniformly bounded, it is concluded that $\lim_{n\to\infty} q_4(k) = q_4^*$ exists with probability one. Further, if $q_4(k) \notin \{0, 1\}$, then $q_4(k + 1) \neq q_4(k)$ with non–zero probability for all k. Hence, $q_4^* \in \{0, 1\}$ constitutes the absorbing set for the Markov process $\{q_4(k)\}$.

Let $q^* \in V_8$ denotes the state to which $q(k)$ converges, where $V_8 = \{e_1, e_2, \ldots, e_8\}$ is the set of all absorbing states for process $\{q(k)\}$. Define

$$\Gamma_4[q] = \text{Prob } \left[q^* = e_4 | q(0) = q\right].$$

Let $C(S_8) : S_8 \to \Re$ be the state space of all real–valued continuously differentiable functions with bounded derivative defined on S_8, where $\Re$ is the real line and $g[q] \in C(S_8)$. The learning algorithm defines an operator U with meaning of $Ug[q] = E\left[g[q(k + 1)]|q(k) = q\right]$, where $E[.]$ represents the mathematical expectation. It can be shown that operator U is linear and preserves positive functions. It has been shown that $\Gamma_4[q]$ is the only continuous solution of $U\Gamma_4[q] = \Gamma_4[q]$, with the following boundary conditions [34].

$$\Gamma_4[e_j] = \begin{cases} 1 & j = 4 \\ 0 & j \neq 4. \end{cases} \tag{12}$$

Proof of theorem 1 for graph of figure 3 : For graph 3, we have 8 paths between the source node and the destination node, i.e. $r = 8$. From the definitions of $c_i(k)$, it is evident that as $L_i(k + 1)$ decreases, $c_i(k)$ also decreases. Since π_4 is the shortest path, we must show that

$$\Delta q_4(k) = E[q_4(k + 1) - q_4(k)|q(k)] > 0 \qquad \forall q_4(k) \in (0, 1). \tag{13}$$

Since π_4 is the shortest path, we have $c_4^* = \min_i\{c_i\}$. For large values of k we have

$$E[q_4(k + 1) - q_4(k)|q(k)] \geq a q_4(k) \sum_{j\neq 4} q_j(k)[c_j(k) - c_4(k)].$$

Therefore $E[q_4(k + 1) - q_4(k)|q(k)] > 0$ for all $q_4(k) \notin \{0, 1\}$. From corollary 1, it is concluded that $q_4^* \in \{0, 1\}$ which constitutes the absorbing set for the process

$\{q_4(k)\}$. Define

$$\Phi_4[x,q] = \frac{e^{-\frac{x}{a}q_4} - 1}{e^{-\frac{x}{a}} - 1}$$

where $x > 0$ is to be chosen. $\Phi_4[x,q] \in C(S_8)$ and satisfies boundary conditions (12). In what follows, we show that $\Phi_4[x,q]$ is sub–regular, thus $\Phi_4[x,q]$ qualifies as a lower bound on $\Gamma_4[q]$. Since super and sub–regular functions are closed under addition and multiplication by a positive constant, and if $\Phi(.)$ is super–regular then $-\Phi(.)$ is sub–regular, it follows that $\Phi_4[x,q]$ is sub–regular if and only if $\Psi_4[x,q] = e^{-\frac{x}{a}q_4}$ is sub–regular. We now determine conditions under which $\Psi_4[x,q]$ is sub–regular. From equation (11), the definition of operator U and some algebraic simplifications, for large values of k, we have

$$\begin{aligned} U\Psi_4[x,q] - \Psi_4[x,q] &= E\left[e^{-\frac{x}{a}q_4(k+1)} \middle| q(k) = q \right], \\ &= xq_4\Psi_4[x,q]G_4[x,q], \end{aligned}$$

where

$$\begin{aligned} G_4[x,q] = -d_4^*(1-q_4)V\left[-x(1-q_4)\right] + \left(\sum_{j\neq 4} m_j d_j^* q_j\right) \\ \times V\left[xq_4\right] + \frac{1}{xq_4}\sum_j q_j(mjd_j^* + c_j^* - 1), \end{aligned} \tag{14}$$

$$V[u] = \begin{cases} \frac{e^u - 1}{u} & u \neq 0 \\ 1 & u = 0 \end{cases} \tag{15}$$

and

$$m_j = \begin{cases} 1 & \text{for } j = 1,2,\ldots,6 \\ e^{\frac{a_3(k)}{a}xq_4(1-a)} & \text{for } j = 7 \\ e^{-\frac{a_3(k)}{a}xq_3(1-a)} & \text{for } j = 8. \end{cases} \tag{16}$$

Since $a_3(k)/a \geq b_1 \geq 1$ and $b_2 \geq a_3(k)/a \geq 1$ for all k and for some b_1 and b_2 we may replace m_7 and m_8 by m_7' and m_8', respectively. That is

$$m_7' = e^{b_1 xq_4(1-a)} \leq m_7$$

and

$$m_8' = e^{-b_2 xq_3(1-a)} \leq m_8.$$

Therefore $\Psi_4[x,q]$ is sub–regular if

$$G_4[x,q] \geq 0. \tag{17}$$

In order to show the above inequality, we first consider the last term in right hand side of equation (14). Since x and q_4 both are non–negative, then we consider only the following summation.

$$\begin{aligned} G_4'[x,q] &= \sum_j q_j[m_j d_j^* + c_j^* - 1] \\ &= \sum_{j=7}^{8} q_j[m_j' d_j^* + c_j^* - 1] \\ &\geq 0. \end{aligned}$$

From m_7' and m_8' and using the Taylor expansion, we have

$$\begin{aligned} m_7' &= 1 + v \\ m_8' &= 1 - u \end{aligned}$$

where

$$\begin{aligned} v &= bq_4 \left[1 + \frac{bq_4}{2!} + \frac{(bq_4)^2}{3!} + \cdots\right] \\ &= bq_4 v_1 \end{aligned}$$

and

$$\begin{aligned} u &= bq_3 \left[1 - \frac{bq_3}{2!} + \frac{(bq_3)^2}{3!} - \cdots\right] \\ &= bq_3 u_1 \end{aligned}$$

and $b = \frac{a_3}{a} x(1-a)$. Thus we have

$$G_4'[x,q] = q_7\left[(1+v)d_7^* + c_7^* - 1\right] + q_8\left[(1-u)d_8^* + c_8^* - 1\right].$$

Therefore, $G_4'[x,q] \geq 0$ if and only if $q_4 q_7 d_7^* v_1 - q_3 q_8 d_8^* u_1 \geq 0$. Thus

$$\frac{q_7 q_4 v_1}{q_8 q_3 u_1} \geq \frac{d_8^*}{d_7^*} \geq 1.$$

By using the definition of q and with some algebraic simplifications we have

$$\frac{v_1}{u_1} \geq \frac{d_8^*}{d_7^*}.$$

So we can find an a_1^* such that for all $a \in (0, a_1^*)$ the condition $\frac{v_1}{u_1} \geq \frac{d_8^*}{d_7^*}$ is satisfied and so inequality (17). Since super and sub–regular functions are closed under addition and multiplication by a positive constant, thus we can drop positive term $G_4'[x,q]$. Therefore, from equation (14), it follows that $\Psi_4[x,q]$ is sub–regular if we have

$$-d_4^*(1-q_4)V\left[-x(1-q_4)\right] + \left(\sum_{j\neq 4} m_j d_j^* q_j\right) V\left[xq_4\right] \geq 0. \tag{18}$$

Define

$$f[x,q] = \frac{V[-x(1-q_4)]}{V[xq_4]} \leq \frac{\sum_{j\neq 4} m_j q_j d_j^*}{d_4^*(1-q_4)}, \tag{19}$$

for all $a \in (0, a_1^*)$. The right hand side of the above inequality consists of nonnegative quantities. Substituting $(1-q_4)$ by $\sum_{j\neq 4}$ in the above inequality, we can write

$$m_8' \min_{j\neq 4}\{\frac{d_j^*}{d_4^*}\} \leq \min_{j\neq 4}\{\frac{d_j^*}{d_4^*}\} \leq \frac{\sum_{j\neq 4} m_j q_j \frac{d_j^*}{d_4^*}}{\sum_{j\neq 4} q_j} \leq \max_{j\neq 4}\{\frac{d_j^*}{d_4^*}\} \leq m_7' \max_{j\neq 4}\{\frac{d_j^*}{d_4^*}\}. \tag{20}$$

Define $H[u] = \ln\ V[u]$. Since $H[u]$ is a convex function, it can be shown that

$$\frac{1}{V[x]} \leq \frac{V[-x(1-q_4)]}{V[xq_4]}, \tag{21}$$

and in view of inequality (19), $G_4[x,q]$ is sub-regular if

$$\frac{1}{V[x]} \leq \frac{\sum_{j\neq 4} m_j q_j \frac{d_j^*}{d_4^*}}{\sum_{j\neq 4} q_j}. \tag{22}$$

It follows that there exists a constant η independent of n such that

$$\eta = m_7' \min_{j\neq 4}\{\frac{d_j^*}{d_4^*}\}.$$

Hence the inequality (21) is satisfied if

$$\frac{1}{V[x]} \leq \eta \qquad 0 < \eta < 1. \tag{23}$$

Since $V[x]$ is continuous and strictly monotonically increasing with $V[0] = 1$, a value of $x = x^*$ exists such that $1/V[x] \leq \eta$ for all $x \in (0, x^*]$. By choosing a value $x = x^*$, inequality (23) and $G_4[x,q] \geq 0$ are hold and $\Psi_4[x,q]$ is sub–regular function, thus

$$\Phi_4[x,q] = \frac{e^{-\frac{x}{a}q_4} - 1}{e^{-\frac{x}{a}} - 1}$$

is sub–regular function satisfying the conditions (12) and hence

$$\Phi_4[x,q] \leq \Gamma_4[q] \leq 1.$$

From the definition of $\Phi_4[x,q]$, we see that given any $\epsilon > 0$ there exists a positive constant $a_2^* < 1$ such that for all positive $a \leq \min\{a_1^*, a_2^*\}$ the inequality

$$1 - \epsilon \leq \Phi_4[x,q] \leq \Gamma_4[q] \leq 1$$

holds. This completes the proof of the theorem.

4. Experiments

To study the feasibility of the proposed algorithms, experiments are conducted on the two following stochastic graphs shown in figures 5 and 6 and are borrowed from 10

- Graph 1 which is shown in figure 5 is a graph with 10 nodes, 23 arcs, $v_s = 1$, $v_d = 10$, and $\pi^* = (1, 4, 9, 10)$. Edge cost distribution is given in table 4.
- Graph 2 which is shown in figure 6 is a graph with 15 nodes, 42 arcs, $v_s = 1$, $v_d = 15$, and $\pi^* = (1, 2, 5, 15)$. Edge cost distribution is given in table 4.

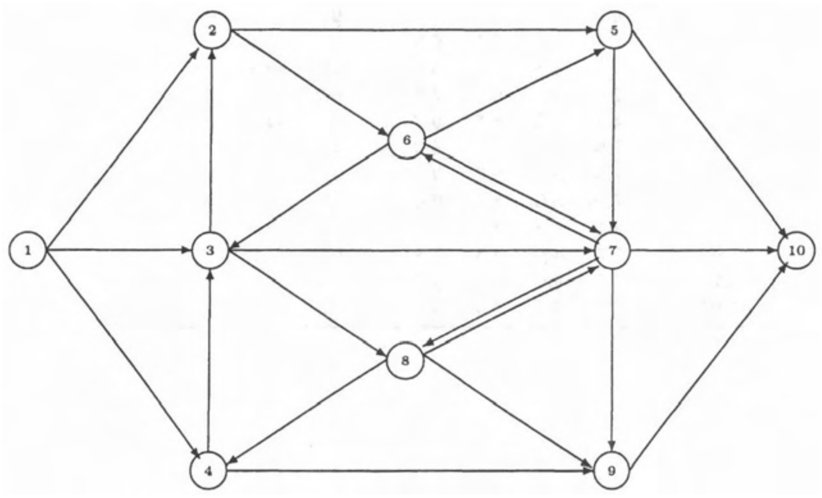

Fig. 5. Graph 1

In all the simulations presented in this paper, the updating scheme for vector probability is L_{R-I}, and each algorithm terminates when the probability of traversed path is 0.9 or 300,000 paths are traversed. Each algorithm is tested on two different graphs and the results for each algorithm are summarized in two tables. Every value in each table is averaged over 100 runs. The first column of the first table includes the average number of iterations required by the algorithm over the runs which it converges. The second column shows the percentage of 100 runs converge to the solution before 300,000 iterations. The second table for each algorithm, shows the total number of samples taken form all the edges in the graph and also the total number of samples taken from the edges along the shortest path. The results of simulations for different graphs are summarized in tables 4 and 4.

From the result of the simulations, the following points can be made.

(1) Simulation results show that the number of converged runs increases as the

Table 1. Weight distribution of graph 1(figure 5)

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

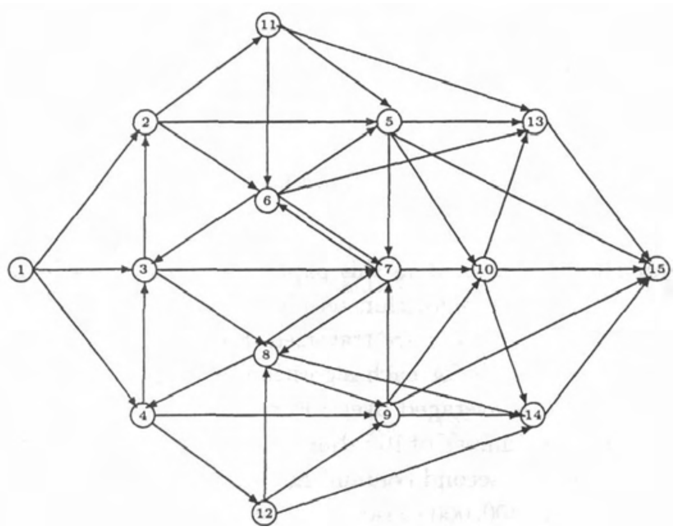

Fig. 6. Graph 2

Table 2. Weight distribution of graph 2 (figure 6)

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	

learning parameter a decreases. For instance for graph 5, 100 of the runs converge when $a = 0.001$,whereas this percentage decreases to 26 when $a = 0.151$ (tables 4 and 4).

(2) Simulation results show that for algorithm 2 the average number of iterations required to reach the termination criteria is lesser comparing to algorithm 1,

Table 3. The simulation results of proposed algorithm for graph 1

a) Average number of iterations and runs converged

	Algorithm 1		Algorithm 2		Algorithm 3		Algorithm 4		Algorithm 5		Algorithm 6	
a	AVI	PC	AVI	PC	AVI	PC	AVI	PC	AVI	PC	AVI	PC
0.001	10911	100	10237	100	34744	100	9734	100	18875	99	49537	100
0.005	2231	100	2121	100	6962	100	1943	100	3589	92	9956	100
0.010	1175	99	1122	100	3424	100	992	100	1662	77	5167	99
0.015	759	97	769	94	2286	100	651	99	1060	80	3668	97
0.020	579	98	536	98	1725	100	497	100	785	72	2758	93
0.025	468	93	446	96	1354	100	408	98	544	71	2140	93
0.030	365	90	352	91	3909	100	312	97	490	71	1692	93
0.035	294	90	298	86	1016	100	271	96	360	65	1432	91
0.040	255	87	287	85	818	100	228	96	253	68	1241	84
0.045	277	86	243	88	770	100	201	92	194	61	1039	82
0.050	227	79	189	84	707	100	166	89	199	59	1010	86
0.055	172	79	167	83	631	100	150	84	202	70	952	89
0.060	167	76	161	81	571	100	148	82	131	63	785	80

b) Average samples taken from graph and optimal path

	Algorithm 1		Algorithm 2		Algorithm 3		Algorithm 4		Algorithm 5		Algorithm 6	
a	TS	SPS	TS	SPS	TS	SPS	TS	SPS	TS	SPS	TS	SPS
0.001	41961	19471	38369	18443	117202	62908	30960	17008	65436	35160	161159	75750
0.005	8548	3870	7914	3752	23490	12571	6212	3361	12870	5967	32362	15396
0.010	4639	1996	4133	1954	11582	6213	3186	1699	6363	2576	16896	7729
0.015	2912	1242	2900	1241	7720	4125	2103	1077	3975	1605	12002	5336
0.020	2195	931	2019	910	5832	3085	1616	795	2875	1152	9385	3924
0.025	1830	722	1670	726	4578	2413	1381	655	2037	767	7156	3072
0.030	1441	565	1332	563	3909	2054	1047	486	1761	675	5663	2441
0.035	1194	474	1161	451	3423	1783	920	426	1492	496	4765	2064
0.040	1019	395	1105	406	2779	1478	788	363	1104	387	4316	1724
0.045	1078	392	953	353	2602	1353	667	313	938	317	3703	1458
0.050	886	313	741	281	2383	1210	605	261	926	290	3435	1401
0.055	765	251	670	257	2136	1107	555	219	801	296	3180	1326
0.060	708	241	658	248	1930	1005	505	204	648	222	2668	1065

but percentage of the number of runs converged is higher. This may be due to the fact that algorithm 2 does not traverse paths containing loops, which leads to smaller number of samples taken from the edges.

(3) Algorithm 3 improves the rate of convergence. This is because of the fact that as we approach the destination node, the action probability vectors are updated more conservatively. On the other hand, the average number of iterations for this algorithm is higher. The reason for this is that the nodes closer to the destination node use smaller step lengths comparing to the nodes closer to the source node..

(4) The updating scheme for step length in algorithm 4 causes degradation in the performance. One reason for such degradation may be that there is no dependency between the step lengths and the action probabilities.

Table 4. The simulation results of proposed algorithm for graph 2

a) Average number of iterations and runs converged

	Algorithm 1		Algorithm 2		Algorithm 3		Algorithm 4		Algorithm 5		Algorithm 6	
a	AVI	PC	AVI	PC	AVI	PC	AVI	PC	AVI	PC	AVI	PC
0.001	20940	100	20302	100	65584	100	15984	100	26385	100	77278	100
0.005	4557	96	4780	98	11034	100	3261	93	6673	96	16637	100
0.010	2489	90	2290	93	6392	100	1732	80	3288	85	7854	100
0.015	1658	91	1600	87	4116	100	1014	78	2179	72	6184	100
0.020	1206	77	1232	82	3495	99	927	63	1449	68	4349	99
0.025	910	79	904	79	2486	98	637	57	1017	48	3649	96
0.030	726	74	765	63	2220	98	521	63	853	56	2664	95
0.035	652	73	582	61	1948	97	503	47	677	43	2550	92
0.040	534	72	548	66	1646	91	352	45	655	39	2485	88
0.045	436	63	468	57	1534	92	315	46	611	38	1727	91
0.050	434	56	392	60	1138	90	218	46	410	40	1808	93
0.055	386	51	319	57	1026	89	218	46	400	37	1475	84
0.060	357	58	346	48	1074	94	192	36	363	32	1359	87

b) Average samples taken from graph and optimal path

	Algorithm 1		Algorithm 2		Algorithm 3		Algorithm 4		Algorithm 5		Algorithm 6	
a	TS	SPS	TS	SPS	TS	SPS	TS	SPS	TS	SPS	TS	SPS
0.001	88230	35193	83697	32541	227099	129590	52662	31435	104269	48384	262317	161110
0.005	23016	9280	19294	7273	45144	24850	11007	5960	25949	10959	56020	33923
0.010	10544	3762	9940	3454	22201	11869	6422	2463	13179	5161	26565	15361
0.015	6811	2498	6520	2208	14357	7345	3784	1421	9793	3368	20630	11186
0.020	5196	1613	5116	1589	11947	6091	3508	1031	6182	2191	14616	7773
0.025	3982	1250	4378	1214	8607	4433	2712	727	4379	1339	11967	6347
0.030	3340	978	3485	896	7630	3781	1931	615	3575	1197	9021	4561
0.035	2819	890	2692	662	6633	3269	1990	498	2949	860	8570	4291
0.040	2365	751	2384	679	5517	2536	1512	359	2579	728	8046	3873
0.045	1927	551	1977	514	5144	2445	1322	342	2231	655	5673	2802
0.050	1995	517	1715	439	3968	1724	1296	255	1769	507	5936	2857
0.055	1723	456	1483	396	3574	1573	1008	231	1702	489	4947	2223
0.060	1540	395	1345	340	3671	1673	861	201	1471	371	4707	2230

(5) Experimentations show that for large value of learning parameter a all algorithms perform the same. Experimentation also show that the average iterations required by the algorithms for finding the optimal path is heavily dependent on the input graph and there is no general ordering according to which the algorithms can be ordered in terms of their performance.

(6) Another property we want to illustrate is *any time behavior* of the proposed algorithm. We show this property by plotting the development of the probability of choosing the shortest path in time (see figure 7). This curve shows rapid progress in the beginning and flattening out later on. This is typical for many algorithms that work by iterative improvements on the initial solution. The name "any time" comes from the property that the search can be stopped at any time and the proposed algorithm will have some solution, even it is suboptimal.

Based on this any time curve, we can make some general observation concerning termination condition for the proposed algorithm. In figure 7, we can divide the run time into two equally long sections, the first and the second half. As this figure indicates, the progress in terms of increments in probability of choosing the shortest path is significantly greater than the achievements in the second half. This provides a general suggestion that it might not be worthwhile to allow very long runs; because of the any time behavior of the proposed algorithm efforts spent after a certain time may not result in better solution quality. A point with coordinate (x, y) in any time curve gives us the average number of iterations x required to obtain the optimal path with probability of y. This is one advantage of the proposed algorithm over non-iterative algorithms.

(7) Probability of optimal path (POP) at each iteration of algorithm is defined as the product of probabilities of edges along the optimal path. The plots of average probability of optimal path over the converged runs out of 100 runs for both graphs and different algorithms are given in figures 7 and 8. A point with coordinate (x, y) in POP curve gives us the average number of iterations required to obtain the optimal path with a specific probability. Looking at different plots indicate the fact that algorithms 3 and 6 perform worst among the algorithms presented in this paper.

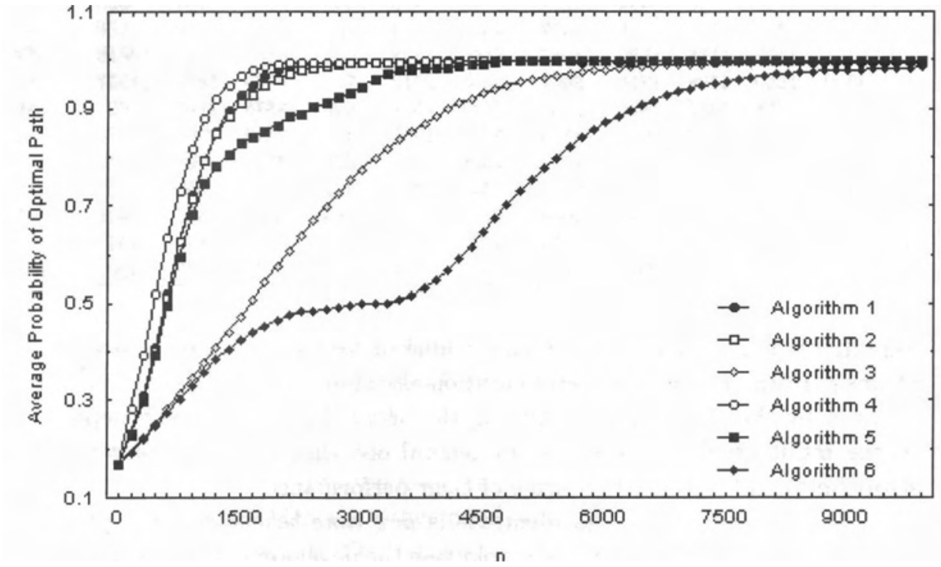

Fig. 7. Probability of optimal path for graph 1

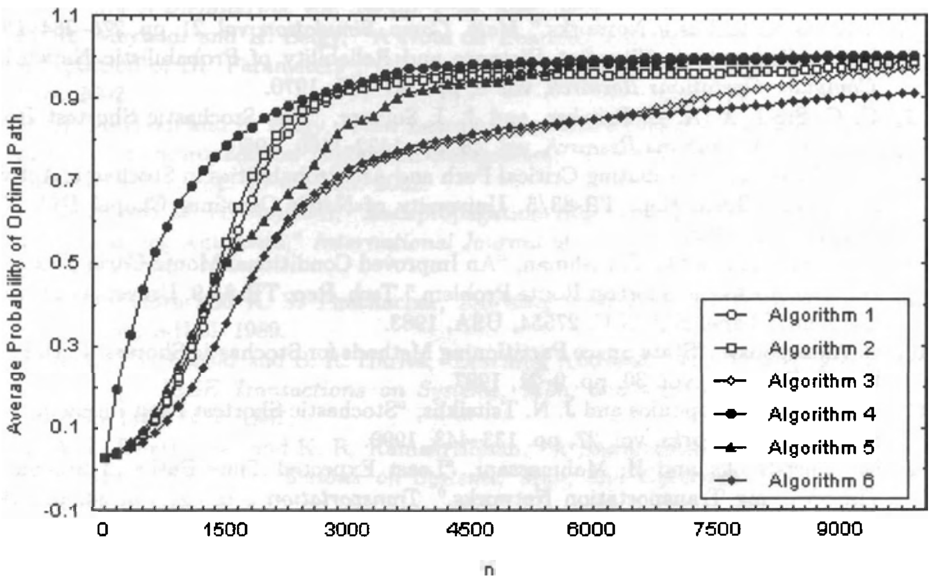

Fig. 8. Probability of optimal path for graph 2

5. Conclusion

In this paper we have studied the problem of determining the optimal path in stochastic networks analogous to deterministic networks. The algorithms presented provide policies which can be used to determine a path from the source to the destination node with minimal expected length. The algorithms presented are adaptive procedures based on distributed learning automata. It is shown that the shortest path is found with a probability as close as to unity by proper choice of the parameters of the proposed algorithms.

Acknowledgements

The authors thank the reviewers for their very helpful comments and suggestions.

References

1. N. Deo, , and C. Pang, "Shortest Path Algorithms: Taxonomy and anotation," *Networks*, vol. 14, pp. 275–303, 1984.
2. A. A. B. Pritsker, "Application of Multi-Channel Queuing Results to the Analysip of Conveyor Systems," *Journal of Industrial Engineeriig*, pp. 14–42, 1966.
3. H. Frank, "Shortest Path in Probabilistic Graphs," *Opertions Research*, vol. 15, pp. 583–599, 1969.
4. J. J. Martin, "Distribution of the Time Through a Directed Acyclic Network," *Operations Research*, pp. 60–56, 1965.

5. C. U. Sizal, V. A. Q. Pritsker, and J. J. Solberg, "The Use of Cutsets in Monte-Carlo analysis of Stochastic Networks," *Math. Comp. Simulation*, vol. 21, pp. 276–384, 1979.
6. P. B. Mirchandani, "Shortest Distance and Reliability of Probabilistic Netwnrks," *Computer Operations Research*, vol. 8, pp. 347–315, 1970.
7. C. C. Sigal, A. A. B. Pritsker, and J. J. Solberg, "The Stochastic Shortest Route Problem," *Operations Research*, vol. 48, pp. 1422–1199, 1980.
8. 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. 57514, USA, 1983.
9. 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. 27554, USA, 1983.
10. C. Alexopoulos, "State Space Partitioning Methods for Stochastic Shortest Path Problems," *Networks*, vol. 30, pp. 9–21, 1997.
11. G. H. Polychronopoulos and J. N. Tsitsiklis, "Stochastic Shortest Path Problems with Recourse," *Networks*, vol. 27, pp. 133–443, 1990.
12. E. Miller-Hooks and H. Mahmassani, "Least Expected Time Paths in Stochastic, Time-Varying Transportation Networks," *Transportation Science*, vol. 34, pp. 198–215, 2000.
13. E. Miller-Hooks, "Adaptive Least-Expected Time Paths in Stochastic, Time-Varying Transportation and Data Networks," *Networks*, vol. 37, pp. 35–52, Jan. 2001.
14. S. T. Waller and A. K. Ziliaskopoulos, "On the Online Stochastic Shortest Paths with Limited Arc Cost Dependencies," *Networks*, vol. 40, pp. 216–227, Dec. 2002.
15. B. J. Oommen and M. Agache, "Continuous and Discretized Pursuit Learning Schemes: Various Algorithms and Their Comparison," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 31, pp. 277–287, June 2001.
16. M. Agache and B. J. Oommen, "Continuous and Discretized Generalized Pursuit Learning Schemes," in *Proceedings of SCI-2000, the 4th World Multiconference on Systemics, Cybernetics, Orlando, USA*, vol. VII, pp. 270–275, July 2000.
17. M. Agache and B. J. Oommen, "Generalized Pursuit Learning Schemes: New Families of Continuous and Discretized Learning Automata," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 32, pp. 738–749, Dec. 2002.
18. P. R. Srikantakumar and K. S. Narendra, "A Learning Model for Routing in Telephone Networks," *SIAM Journal of Control and Optimization*, vol. 20, pp. 34–57, Jan. 1982.
19. O. V. Nedzelnitsky and K. S. Narendra, "Nonstationary Models of Learning Automata Routing in Data Communication Networks," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-17, pp. 1004–1015, Nov. 1987.
20. B. J. Oommen and E. V. de St. Croix, "Graph Partitioning Using Learning Automata," *IEEE Transactions on Computers*, vol. 45, pp. 195–208, Feb. 1996.
21. M. R. Meybodi and H. Beigy, "Solving Stochastic Shortest Path Problem Using Distributed Learning Automata," in *Proceedings of 6th Annual Iran Computer Society of Iran Computer Conference CSICC-2001, Isfehan, Iran*, pp. 70–86, Feb. 2001.
22. H. Beigy and M. R. Meybodi, "Solving the Graph Isomorphism Problem Using Learning Automata," in *Proceedings of 5th Annual International Computer Society of Iran Computer Conference, CISCC-2000, Tehran, Iran*, pp. 402–415, Jan. 2000.
23. B. J. Oommen and T. D. Roberts, "Continuous Learning Automata Solutions to the Capacity Assignment Problem," *IEEE Transactions on Computers*, vol. 49, pp. 608–620, June 2000.
24. M. R. Meybodi and H. Beigy, "Neural Network Engineering Using Learning Automata: Determining of Desired Size of Three Layer Feedforward Neural Networks," *Journal*

of Faculty of Engineering, vol. 34, pp. 1–26, Mar. 2001.

25. M. R. Meybodi and H. Beigy, "A Note on Learning Automata Based Schemes for Adaptation of BP Parameters ," *Journal of Neurocomputing*, vol. 48, pp. 957–974, Nov. 2002.
26. M. R. Meybodi and H. Beigy, "New Learning Automata Based Algorithms for Adaptation of Backpropagation Algorithm Parameters," *International Journal of Neural Systems*, vol. 12, pp. 45–68, Feb. 2002.
27. H. Beigy and M. R. Meybodi, "Backpropagation Algorithm Adaptation Parameters using Learning Automata," *International Journal of Neural Systems*, vol. 11, pp. 219–228, June 2001.
28. K. S. Narendra and K. S. Thathachar, *Learning Automata: An Introduction.* New York: Printice-Hall, 1989.
29. M. A. L. Thathachar and B. R. Harita, "Learning Automata with Changing Number of Actions," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-17, pp. 1095–1100, Nov. 1987.
30. M. A. L. Thathachar and K. R. Ramakrishnan, "A Heirarchical System of Learning Automata," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-11, pp. 236–248, Mar. 1981.
31. A. M. Mood, F. A. Grabill, and D. C. Bobes, *Introduction to the theory of statistis.* McGraw-Hill, 1963.
32. S. Lakshmivarahan and M. A. L. Thathachar, "Bounds on the Convergence Probabilities of Learning Automata," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, pp. 756–763, Sept. 1976.
33. M. R. Meybodi and S. Laksmivarahan, "A Learning Approach To Priority Assignment In A Two Class M/M/1 Queueing With Unknown Parameters," in *Proceedings of The Third Yale Workshop on Applications of Adaptive Systems Theory*, pp. 106–109, June 1983.
34. M. F. Norman, *Markovian Process and Learning Models.* New York: Academic Press, 1972.