A Novel Self-adaptive Search Algorithm for Unstructured Peer-to-Peer Networks Utilizing Learning Automata

Mahdi Ghorbani
Dept. of computer engineering and information technology engineering
Qazvin Branch, Islamic Azad University
Qazvin, Iran
m.ghorbani@qiau.ac.ir

Mohammad reza Meybodi, Ali mohammad Saghiri
Dept. of computer engineering
Amirkabir University of Technology
Tehran, Iran
mmeybodi@aut.ac.ir, a_m_saghiri@aut.ac.ir

Abstract— **Designing an efficient search algorithm is an important issue in unstructured peer-to-peer networks when there is no central control or information on the locations of objects. There are various search strategies with different effects on network performance. In k- random walks as a search strategy, having an adaptive value of k instead of a random value can affect performance of the network. Therefore in this paper, a distributed novel self-adaptive search algorithm has been developed by application of learning automata to overcome this challenge. This method does not aim to determine the value of k for k-random walks algorithm and each peer can issue walkers in a self adaptive manner. Simulation results show that the proposed search algorithm improves some features such as average number of walkers per query, average number of produced messages, number of hits per query and also success rate efficiently in comparison with the k-random walks algorithm.**

Keywords-Unstructured peer-to-peer; searching; k-random walks; learning automata

I. INTRODUCTION

Peer-to-peer networks have been growing rapidly in the last few years. Many applications of the peer-to-peer systems like file sharing are due to the distributed architectures of these networks. Peer-to-peer networks are usually implemented with no central control in order to obtain some features such as flexibility and scalability. Peer-to-peer networks can be categorized into two classes: structured and unstructured. In structured peer-to-peer networks, placement of the contents is controlled via Distributed Hash Table (DHT) which makes a small overhead on the network. Placement of the contents in unstructured peer-to-peer networks is done more loosely, where nodes can join and leave without a strict control [1-2]. Therefore, applying a search procedure cannot locate a node directly and it is necessary to use search mechanisms. Nowadays, most applications of peer-to-peer systems focus on unstructured peer-to-peer networks and designing an efficient search method is the most important issue [2]. Search methods in unstructured peer-to-peer networks can be divided to two groups: Blind and Informed [3-4]. In blind search strategies, nodes keep no information about placement of the objects. They rather use flooding techniques to forward the queries. On the other hand, in informed approaches, nodes are aware of network status and placement of the contents by storing some metadata of information. K-random walks algorithm [4-6] is an instance of blind search method. When an attempt to locate an object is unsuccessful, k nodes will pick among the neighbors randomly and the queries are forward to them. The search is successful once the object is found by any one of the individual random walks [6].

The optimum value of k and adaptive to network size improves performance of the network in terms of network load and response time. Selection of a value for k larger than the average number of neighbors, results in a more traffic in the network, because it works like flooding search methods [3-4]. On the other hand, if k is smaller than the average number of neighbors it will decrease the probability of selecting desirable neighbors to locate the objects. Having assigned a proper value for k, it is now important which number of k neighbors should be selected among the whole neighbors.

In some recent studies [7-8][13-19], reinforcement learning techniques [9-10] such as Q-learning [10] are utilized to improve search features. By applying learning mechanisms in a network, each node can learn about network status and make decision for the next iteration of search based on feedbacks from the previous nodes. Learning-based search approaches are expected to make shorter search response time, smaller load of network and improve bandwidth usage.

In this paper, a novel efficient adaptive search algorithm based on learning automata (LA) [11-12] is proposed in unstructured peer-to-peer networks for selecting the best neighbors to obtain favorable resources. Selection issues of non-adaptive value of k in k-random walks algorithm are solved and the suggested algorithm is evaluated via several simulations.

978-1-4673-6315-0/13/$31.00

The reminder of this paper is organized as follows. The related work is reviewed in Section II. Section III, describes Learning Automata as a reinforcement learning technique. In section IV, the proposed algorithm is introduced. Section V discusses the simulation methodology, while Section VI concludes the paper.

II. RELATED WORKS

As previously discussed, search strategies in unstructured peer-to-peer networks are classified in two groups of blind and informed methods.

In the latter, each node maintains some information about its neighbors and network status. APS [5][7], DST [13-14] and ISA [19] are some examples of the informed methods where user employs past experience of the neighbor to search the objects for the next iterations. Also in [15] and [16], a novel search algorithm for peer-to-peer networks is introduced which uses a new form of machine learning technique to store some information about the neighbors.

Adaptive Probabilistic Search (APS), relies on utilizing k independent walkers and probable forwarding. Each intermediate node, forwards the query to its neighbor with the probability value stored in its local index. Index values are updated by receiving feedback from the walkers. APS increases reliability and improves bandwidth consumption but the overhead is still considerable.

Distributed Search Technique (DST), uses Q-learning method to select the favorable neighbors in order to forward the queries. In this method, all nodes are grouped into ordinary and power nodes. At first, k walkers are propagated through the network and the queries are forwarded to the nodes to locate the objects. If the search seems unsuccessful, propagation of the queries is performed through both ordinary nodes and power nodes based on the past experience in memories of the nodes. It is interesting to note that the search performance relies on the progress of learning which is sometimes very slow.

Intelligent Search Algorithm (ISA) applies learning automata in order to learn network status and location of the objects in the network. This method works same as DST except that each peer selects k walkers with the highest p-value within its experience tables. In ISA each peer randomly selects the value of its tables in different iterations, such that almost the nodes will participate in the selection process during each iteration. In some iterations, the neighbors with the probability of having the objects, are forgotten. Therefore the success rate will be decreased.

In [15] and [16], an adaptive version of k-random walks algorithm for peer-to-peer networks is introduced. The suggested algorithm is based on adaptive select of walkers according to each node's feedback. The proposed search algorithm considers the feedbacks from the environment and then makes decision about suitable nodes in order to participate in search. The decision making for selecting neighbors is a new form of learning automata which is named KSALA [18]. Due to intelligent technique, the learning rate for selecting the neighbors is slow.

In blind methods, each node forwards the query to all of its neighbors. Search is completed when "hit" or "miss" is occurred or when TTL is terminated. In some of these strategies such as k-random walks [6], the query is routed to some of the nodes which are randomly selected instead of forwarding them to all the neighbors. Applying these search methods will reduce performance of the network due to randomly forwarding the queries and lack of an adaptive solution when the network load is unsteady [5-6].

III. OVERVIEW OF LEARNING AUTOMATA

A learning automaton [11-12] is an adaptive decision-making system that can improve its performance by learning how to choose the optimal action from a set of allowed actions through repeated interactions with the random environment. At each iteration, the selected action is served as the input to the random environment. The environment responds the taken action in turn with a reinforcement signal. The action probability vector is updated based on the reinforcement feedback from the environment. The objective of a learning automaton is to find the optimal action from the action-set so that the average penalty received from the environment is minimized.

The environment can be described by a triple $E \equiv \{\alpha,\beta,c\}$, where $\alpha \equiv \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ denotes the finite set of inputs, $\beta \equiv \{\beta_1, \beta_2, \ldots, \beta_m\}$ is the set of the values that can be taken by the reinforcement signal, and $c \equiv \{c_1, c2, \ldots, c_r\}$ denotes the set of the penalty probabilities. Depending on the nature of the reinforcement signal b, environments can be classified into P-model, Q-model and S-model. The environments in which the reinforcement signal can only take two binary values 0 and 1 are referred to as P-model environments. Another class of the environment allows a finite number of the values in the interval [0,1] can be taken by the reinforcement signal. Such an environment is referred to as Q-model environment. In S-model environments, the reinforcement signal lies in the interval $[a,b]$. The relationship between the learning automaton and the random environment is shown in fig. 1.

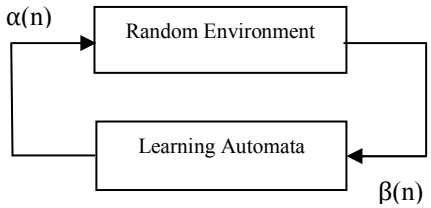

Figure. 1. The relationship between the learning automaton and its random environment [12].

Learning automata can be classified into two main families [12]: fixed structure learning automata and variable structure learning automata. Variable structure learning automata are represented by a triple $<\underline{\beta},\underline{\alpha},L>$, where $\underline{\beta}$ is the set of inputs, $\underline{\alpha}$ is the set of actions, and L is learning algorithm. The learning algorithm is a recurrence relation which is used to modify the action probability vector. Let $\alpha_i(k) \in \alpha$ and $\underline{p}(k)$ denote the action selected by learning automaton and the probability vector defined over the action set at instant k, respectively. Let a and b denote the reward and penalty parameters and determine the amount of increases and decreases of the action probabilities, respectively. Let r be the number of actions that can be taken by the learning automaton. At each

instant k, the action probability vector $\underline{p(k)}$ is updated by the linear learning algorithm given in (1), if the selected action $\alpha_i(k)$ is rewarded by the random environment, and it is updated as given in (2) if the taken action is penalized.

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

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

If $a = b$, the recurrence (1) and (2) are called linear reward penalty (L_{RP}) algorithm, if $a \gg b$ the given equations are called linear reward-ε penalty ($L_{R\varepsilon P}$), and finally if $b = 0$, they are called linear reward-Inaction (L_{RI}). In the latter case, the action probability vectors remain unchanged when the taken action is penalized by the environment.

IV. DESIGN

We focus on unstructured peer-to-peer networks and apply learning automata in order to train nodes during search procedure. In this paper, we use S-model environment and LA is L_{RP}. First, data structure of the proposed algorithm is described, and then, the search algorithm is presented. As mentioned before in section III, probability values are much important to make decisions in LA and are thus updated according to reward/penalty feedbacks from the environment.

To apply LA training nodes for making decision about the search in different iterations, one will need to become familiar with some simple definitions. Suppose current node to which the queries are forwarded as P_{Vector} while the probability values are shown as P_{ij}. Therefore, one may consider the following descriptions:

$$N = \{N_1, N_2, \ldots | N_i \text{ is the } i^{th} \text{ neighbor of CN}, i = 1,2,\ldots\} \quad (3)$$

$$P_{Vector} \quad (4)$$
$$= \{P_{1i}, P_{2i}, \ldots, P_{ri} | \ r \text{ is the number of actions for } N_i, \ \sum P_{ki} = 1\ \}$$

In the proposed algorithm, each node utilizes an LA algorithm and keeps a table of accessible objects (AOs) for each neighbor in each column and row which keeps neighbor-ID and its . As it is shown in fig. 2, P_{Vector} of each neighbor has two probability values of P_{1i} and P_{2i} which are defined as below:

"P_{1i} is N'_i s probability value for participation in previous search iterations while P_{2i} represents N'_i s probability value which is not participated in past iterations of search." The values of P_{Vector} are initialized by 0.5.

	AO_1
N_1	$\{P_{11}, P_{21}\}$
N_2	$\{P_{12}, P_{22}\}$
N_3	$\{P_{13}, P_{23}\}$
⋮	
N_r	$\{P_{1r}, P_{2rs}\}$

Figure 2. A simple format of ***CN***'s table for AO_1

For the purpose of learning in this algorithm the tables are categorized in three groups: "Neighbor-hit-LA-table", includes the neighbors in which the previously forwarded query has produced "hit" based on the previous search steps. Second table is called "Neighbor-miss-LA-table" which includes neighbors participated in search during the past iterations of searching an object though the result has been appeared to "miss". This table is formed like the Neighbor-hit-LA-table. The rest of neighbors which has not participated in the previous search are stored in "Query-LA-table". It is obvious that in this case, there is no accessible object for each neighbor in Query-LA-table.

Every time a node forwards a query to the neighbors, a new row is added to the table. If the object is not found, locating the object will be continued via tables. First, Neighbor-hit-LA-table will be checked whether the query keyword exists or not. When the query keyword is present in the table, all of the neighbors having the highest probability value according to the relevant P_{Vector} are selected to keep on the search. Having located the object using the selected neighbors, all the values are updated based on rewards or penalties in the LA algorithm. If the query keyword is absent in the Neighbor-hit-LA-table, other nodes with the highest probability value in the Query-LA-table will be candidate to locate the objects. Finally, for an unsuccessful search by applying the two mentioned tables, the neighbors from Neighbor-miss-LA-table will be selected regarding the highest probability.

To update the probability values in order to evaluate rewards/penalties in this search algorithm, (1) and (2) are applied in section III. As mentioned before, S-L_{RP} algorithm is used and two actions are considered for LA. Figures 3 and 4 depict updating the probability vector and the proposed search algorithm, respectively.

***/CN ≡ Current node selected for search, N ≡ Neighbor of CN**
P_{vector} ≡ Set of probability values for N /*

If CN participates in search *//according to past iteration*
  if the feedback of Ns is "HIT" *//at least one of the feedbacks is HIT*
    then EQ(1) updates the P_{vector}
  else EQ(2) update the $P_{vectors}$
else *//CN has no participation in search*
  if all the replies of Ns is "MISS"
    then EQ(1) updates the P_{vector} *//persuading Ns to be applied in next iterations*
  else EQ(2) updates the P_{vector} *//decreasing the probability values of non-profitable selections*

Figure 3. Updating $P_{vectors}$ according to feedbacks

/ Q ≡ Query Keyword P_c ≡ Probability values of participation in search, M ≡ Total number of neighbors with the highest P_C /

1. User submits a query
2. Search query node for Q
3. If Q is not in CN
  3.1. Search for Q in the Neighbor-hit-LA-table
    3.1.1. If Q is found
      Select all neighbors with the highest P_C

Generate M query messages
Search starts with M neighbors
Else
Select M neighbors in the Query-LA-table
Generate M query messages
Search starts with M neighbors
Else
Select M neighbors in the Neighbor-miss-LA-table
Generate M query messages
Search starts with M neighbors
4. If a hit occurs, sent back results on the reverse path
5. All nodes on the path update appropriate LA-table with learning automata algorithm

Figure 4. Proposed search algorithm

V. SIMULATION

In this section we describe the simulation environment and present the results of the experimental evaluation of our algorithm. Subsection V-A describes the simulation parameters and their default values to implement the algorithm. Subsection V-B is about the results of the experiments and their evaluations.

A. *Simulation setup*

OverSim [20] is utilized to simulate the search algorithm and evaluate the experiments.

The network model is described first and then the required parameters of simulation are set accordingly. The network is constructed using a random graph [7] with different numbers of nodes for different simulations. The average out-degree of each node is 3. There are 100 distinct objects which are randomly distributed in various nodes. The maximum time to deliver messages of the queries is called TTL and is assumed to be 6. The rate of node failure is 20% which indicates the ratio of inactive nodes to total nodes. The simulations are performed in twenty rounds with each round containing different numbers of walkers. "One round" means "running the protocol" on all the nodes at the same time. Table I summarizes simulation parameters with their default values.

TABLE I. SIMULATION PARAMETERS

Parameters	Default values
Network Topology	Random Graph
Network Type	Unstructured
No. of Nodes	Variable
No. of Objects	100
Time To Live	6

B. *Experimental evaluation*

Several simulations were performed considering network metrics and quality of the search results in order to measure efficiency of the proposed algorithm. The produced search algorithm was compared with k-random walks algorithm, Adaptive Probabilistic Search (APS) and Distributed Search Technique (DST). The following experiments and graphs show our results by performing the simulations in twenty rounds of the mentioned algorithms on this network

1) *Different Churn Rates*

Due to the dynamic network used, node failures and link failures have a significant effect on success rate of the search algorithm. To confirm this, the simulations were run using a 1000 node network assuming node failure rate and link failure rate of 20%. Figure 5 illustrates the success rate for the proposed algorithm with the mentioned conditions. As it is shown in Figure 5, simultaneous node failure and link failure cause a rather low success rate in comparison with node failure or link failure alone. This observation is attributed to missing favorable neighbors and losing routes of the forwarding queries. If a node fails or leaves the network, it is possible to forward a query via other links to substitute that node.

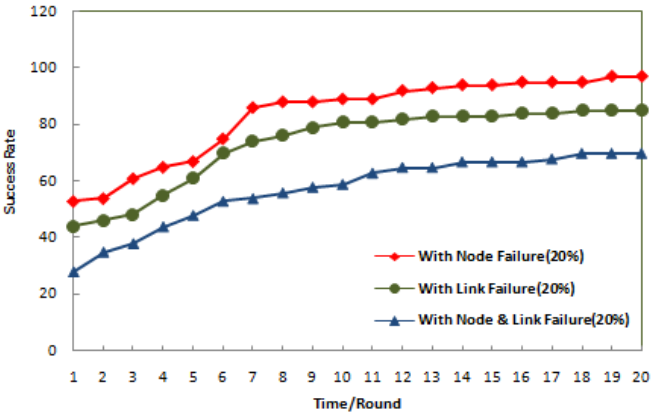

Figure 5. Success rate of our proposed algorithm under different churn

2) *Scalability*

In order to study network scalability, the network was constructed using a random graph comprising of 2000 nodes with 20% node failure. Figure 6 shows that increasing the number of nodes has no further effect on the success rate in the proposed algorithm and the average success rate is obtained about 84%. The recently proposed algorithm has a high performance since the neighbors are selected more accurately.

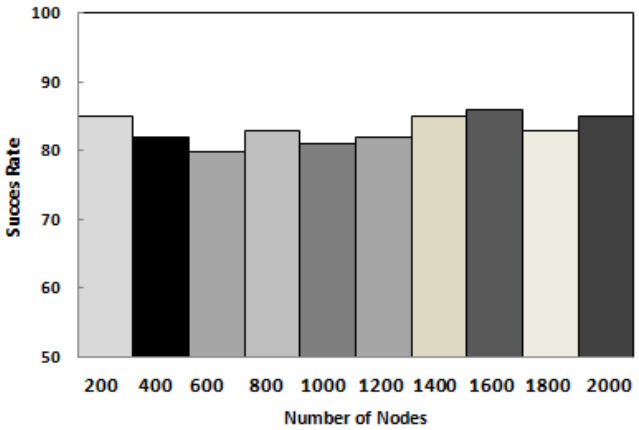

Figure 6. Effect of scalability on success rate under our proposed algorithm

3) *Success Rate*

Applying a network consisting of 1000 nodes with 20% node failure rate, it is shown in Figure 7 that the developed search algorithm has a high success rate after 5th round of running the algorithm. The average search success rate for this algorithm is about 84%, while DST and APS are 83% and 80%, respectively. In k-random walks due to the random selection of the walkers, a rather low success rate of about 52% is expected. Using tables with the priority of selecting suitable neighbors is the main reason for the high success rate of this algorithm. In the worst case, all the neighbors are selected to perform the search but this work is done according to the priority of the groups of neighbors.

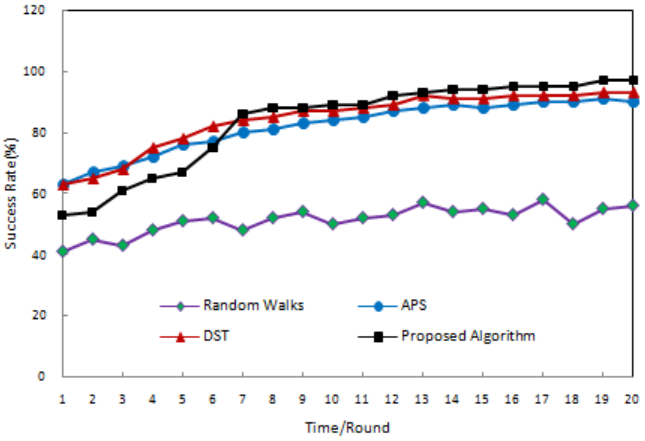

Figure 7. Success rate of four algorithms under a network comprising of 1000 nodes

4) *Average Number of Walkers per Query*

For each round of running the mentioned algorithm, an average number of the selected walkers in each round will be calculable. At the earlier rounds of this algorithm, walkers are selected randomly because of the slow learning phase. Therefore, it is obvious that the average number of walkers will be high at the first steps. These results are shown in Figure 8 after 4th round which demonstrate that the average number of the selected walkers decreases until it converges a stable value. In k-random walks algorithm, more execution rounds cause increasing of the average selected walkers, but the simulation shows that its rate is rather low.

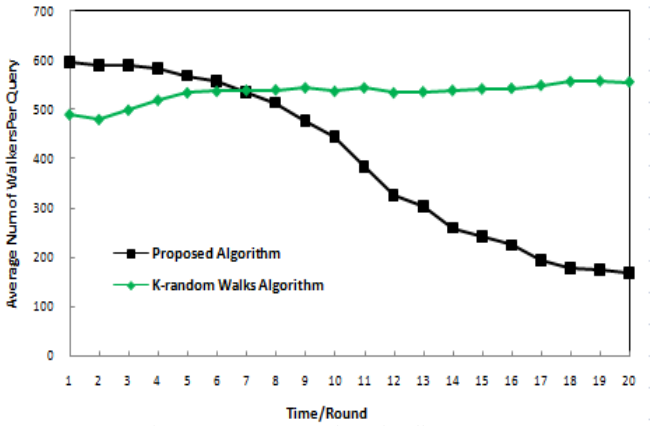

Figure 8. Average number of walkers per query

5) *Overhead of Different Algorithms*

Figure 9 depicts a comparison between the algorithm and four different algorithms related to the messages produced per query. Time to live is 6 and it can control the produced messages. It can be seen that due to the intelligent selection of the neighbors in the developed algorithm, the average number of the messages produced in twenty rounds of simulation is low. APS and DST each generate a great number of messages during the search procedure, while the k-random walk algorithm demonstrates a high average number of the produced messages per query because of its random behavior.

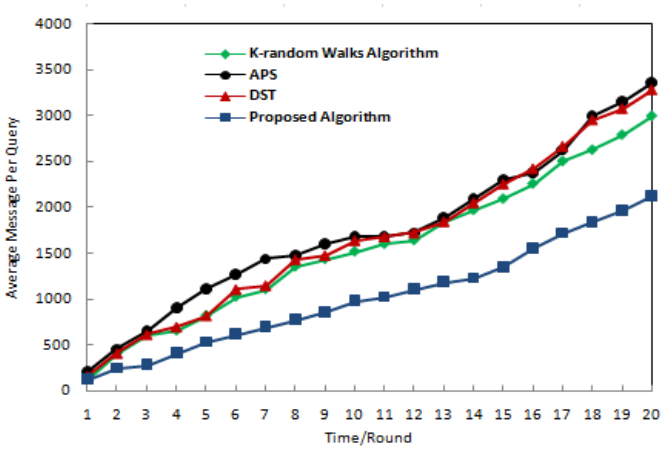

Figure 9. Produced messages per query under four algorithms

6) *Discovered Objects per Query*

Each "hit" message means that an object is found. The suggested algorithm provides more precise results than DST, APS and k-random walks. Since in this algorithm the selected walkers always have the highest probability of participation in search, it will increase the chance of discovering objects when the query is propagated.

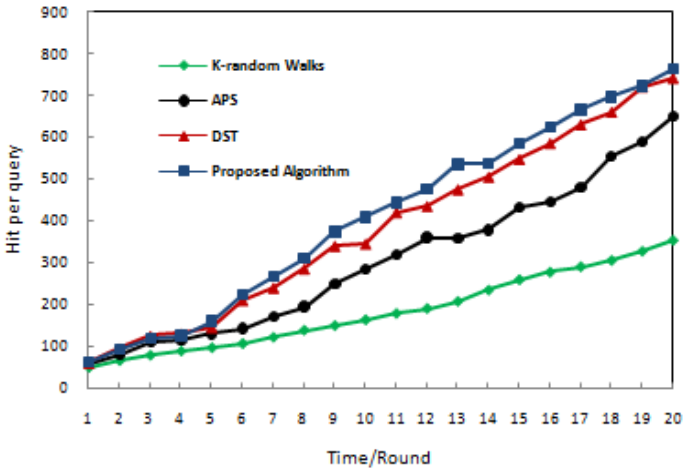

Figure 10. Hit per query under four algorithms

7) *Search Delay*

Figure 11 compares the delay different algorithms need to discover the objects. The delay is based on average number of hops visited for a search. A smaller hop number means less delay and more speed in discovering the objects. This algorithm performs better than APS and k-

random walks though DST is faster than other algorithms. Although the learning phase of the proposed algorithm and updating P_{Vector} is accurate but it is sometimes slow.

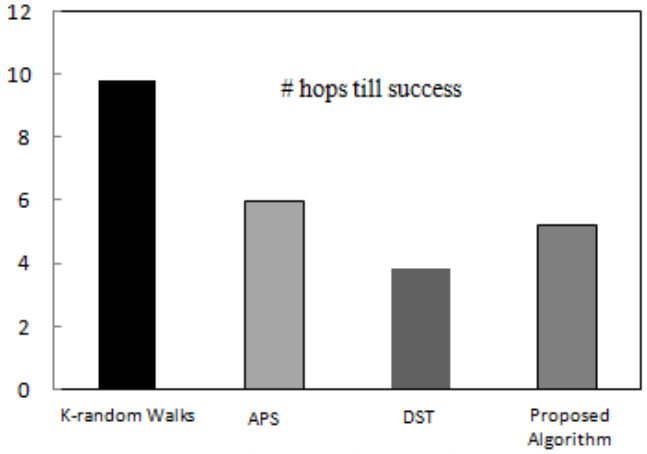

Figure 11. Delay comparison

VI. CONCLUSION

In this paper a self-adaptive search algorithm is introduced for peer-to-peer networks using learning automata. Our proposed algorithm is based on adaptive selection of the walkers according to feedback of each node. The proposed search algorithm solved the challenge of selecting random value of k in k-random walks algorithm. By applying LA for each node in the network, all the neighbors with the highest probability of successful search in the past iterations are selected adaptively to continue the search. Performance of this algorithm was compared with k-random walks algorithm, APS and DST via simulations. In these simulations network metrics such as scalability and average number of hops were considered and some results based on the quality of search mechanisms such as success rate, average number of walkers per query and average number of produced messages were addressed. The simulation results revealed that this new search algorithm can improve success rate, average number of walkers, number of hits per query and average number of messages produced in comparison with other methods. They also provide a shorter delay than k-random walks and APS.

REFERENCES

[1] S. Androutesellis, and D. Spinellis, "A survey of peer-to-peer content distribution technologies," ACM Computing Surveys, vol. 36, no. 4, pp. 335-371, December 2004.

[2] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network scheme," IEEE Communication Survey and Tutorial, March 2004.

[3] D. Tsoumakos, and N. Roussopoulis, "Analysis and comparison of p2p search methods," 1st Int. Conf. Scalabale Information Systems, Article no. 25, 2006.

[4] S. M. Thampi, C. K. Sekaran, "Survey of search and replication schemes in unstructured p2p networks," Network Protocols and Algorithms, vol 2, no. 1, pp. 93-131, 2010.

[5] R. Dorrigiv, A. L'opez-Ortiz and P. Pralat, "Search algorithms for unstructured peer-to-peer networks," In Proceeding of 32nd IEEE Conference on Local Computer Networks, pp. 343-349, 2007.

[6] C. Gkantsidis, M. Mihail, and A. Saberi, "Random walks in peer-to-peer networks," In INFOCOM, Hong Kong, vol. 1, pp. 120-130, 2004.

[7] D. Tsoumakos and N. Roussopoulos, "Adaptive probabilistic search for peer-to-peer networks," 3rd Int. Conf. P2P Computing, pp. 102-109, 2003.

[8] D. Tsoumakos and N. Rossopoulos, "Probabilistic knowledge discovery and management for p2p networks," P2P Journal, 2003.

[9] R. S. Sutton, A. G. Barto, "Reinforcement learning: introduction," In Proceeding of the MIT Press, 1996.

[10] E. Mance and S. H. Stephanie, "Reinforcement learning: A tutorial," In Proceeding of the Wright Laboratory, 1996.

[11] K. Najim, and A. S. Poznyak, "Learning automata: theory and application," in Proceeding of the Tarrytown, New York, Elsevier Science Publishing Ltd., 1994.

[12] K. S. Narenda, and M. Thathachar, Learning Automata: an introduction, New York, Prince-Hall, 1989.

[13] S. M. Thampi, and C. K. Sekaran, "An efficient distributed seach technique for unstructured peer-to-peer networks," Int. Jou. Computer and Network Security, vol. 8, no. 1, pp. 128-135, January 2008.

[14] S. M. Thampi, and C. K. Sekaran, "Collaborative load-balancing scheme for improving search performance in unstructured p2p networks," In Proceeding of the first Int. Conf. Contemporary Computing, pp. 161-169, August 2008.

[15] M. Ghorbani, A. M. Saghiri, and M. R. Meybodi, "A Learning Automata based Adaptive Search Method for Unstructured Peer-to-Peer Networks," In Proceeding of 18th National Conference of Computer Society of Iran, Tehran, Iran, March 14-16, 2013.

[16] M. Ghorbani, A. M. Saghiri, and M. R. Meybodi, "A Learning Automata based Random Walk Search algorithm for Peer-to-Peer Networks," In proceeding of 11th Iranian Conference on Intelligent Systems, Tehran, Iran, February 27-28, 2013.

[17] M. Ghorbani, A. M. Saghiri, and M. R. Meybodi, "A Novel Learning based Search Algorithm for Peer-to-Peer Networks," Technical Journal of Engineering and Applied Science, vol. 3, no.2. ,pp. 145-149, 2013.

[18] S. M. Abolhasani, and M. R. Meybodi, "LADIT: Learning Automata Based Protocol for Routing in Sensor Networks," 2th Conference on Sensor Networks, Yazd, pp. 20-33, 2008.

[19] F. Torabmostaedi, and M. R. Meybodi, "An Intelligent Search Algorithm Based on Learning Automata for Peer-to-Peer Networks", Int. Conf. Contemporary Issues in Computer and Information Sciences, Zanjan, pp. 495-500, June 2011.

[20] I. Baumgart, and B. Heep, Oversim community site, [Online], Available: http://www.oversim.org/wiki