A New Version of K-random Walks Algorithm in Peer-to-Peer Networks Utilizing Learning Automata

Mahdi Ghorbani
Dept. of electrical, computer and IT 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— **One of the most important issues in peer-to-peer networks is locating objects among a lot of data. There have been different search methods to find data with certain advantages and disadvantages. In this paper, we propose a new version of k-random walk algorithm utilizing learning automata. In this distributed method, the value of k for k-random walk is not selected randomly but it is selected in an adaptive manner. It is decided which walker is more useful to be selected in order to keep on the search according to past experience of each node. Simulation results show that the novel search algorithm improves the number of hits per query, success rate, generated messages per query and objects discovery delay in comparison with the k-random walk algorithm.**

Keywords-peer-t-peer; search; random walk; learning automata theory

I. INTRODUCTION

Peer-to-peer networks have been considered as dynamic and flexible networks since emergence of the computer networks for the first time. Arrival of the internet to homes has made the peer-to-peer networks usable by some new and special applications including news broadcasting and sharing. Nowadays, more than 60% of traffic on the internet is owing to these networks [1]. The main characteristic of them is that the nodes can be connected to the network or leave it at any time. In other words, there is no central control on behavior of the nodes in the network, with all the nodes acting similarly in terms of performance level. One of the important issues in these networks is searching stored objects in the nodes. Finding an object will be exposed to some challenges with respect to how the nodes are located in the network. The peer-to-peer networks are divided into structured and unstructured regarding their structures. In the former, e.g. CAN [2], CHORD [3] and Pastry [4], location of the nodes is predefined by distributed hash tables (DHT) so finding an object is done readily using hash functions. On the other hand, the latter, e.g. Gnutella [5] and Napster [6] distribute their contents in a completely random fashion and the nodes have no information about the network status and location of the stored objects in other nodes [1]. Therefore, it is not simple to find an object in this structure and search mechanisms must be utilized here. Since most applications of the peer-to-peer networks refer to unstructured networks, it is very important to design a search method for them and it can further affect efficient of the network considerably.

Search methods in peer-to-peer networks are categorized based on different criteria. They are divided into two groups, namely deterministic and probabilistic regarding how the search queries are directed [7]. In the deterministic methods, previous information is used in the search path for routing, with the direction of queries being predefined. However, the probabilistic methods send the queries both randomly and based on the probabilistic values stored for each node [8]. Based on information of the nodes from the contents, the search techniques are classified as informed and blind [7-9]. In informed methods, the nodes store a number of metadata from their neighborhood. By this kind of information, the nodes would be informed about the network status as well as the location of contents for the other nodes. In blind search methods, the nodes have no information about location of the objects, so they employ flooding algorithms to direct their search queries. One of these methods is random walk [11-13]. Once a search for finding an object is unsuccessful, random walk selects some of the neighbors randomly and sends the search queries to them through a flooding algorithm until the search time is terminated. Each of these two search methods (informed and blind) has its own advantages and disadvantages which can noticeably affect some network criteria such as search success, average response time, average number of objects found in the search process and network load.

In some recent studies [14-19][23], reinforcement learning techniques [20-21] such as Q-learning [20] are utilized to improve search features. Applying learning mechanisms in a network, each node can learn about network status and makes decision for next iteration of search based on previous nodes' feedbacks. It is expected that learning-based search approaches make better search response time, decrease load of network and affect the usage of bandwidth efficiently.

In this paper, we propose a new version of k-random walk algorithm based on learning automata in unstructured peer-to-peer networks to choose the appropriate neighbors

in order to find the resources. Our proposed algorithm, improves the success rate and number of hits per query in comparison with k-random walk algorithm. We also evaluate our algorithm via simulation.

The rest of this paper is organized as follows. The related works is reviewed in Section II. Section III, describes Learning Automata as a reinforcement learning techniques. In section IV, our algorithm is presented. Section V discusses the simulation methodology. Section VI, concludes the paper.

II. RELATED WORKS

As discussed before, search strategies in unstructured peer-to-peer networks based on node's awareness of network status, are classified in the groups of blind and informed methods. In blind methods, each node forwards the query to all of its neighbors. Search is finished when "hit" or "miss" occurred or TTL is terminated. Blind search methods waste the network bandwidth and yield much overhead [10]. In some of these strategies such as k-random walks, instead of forwarding the query to all neighbors, it is routed to partial of nodes which are randomly selected. Applying these search methods decrease performance of the network due to forwarding queries randomly and lack of an adaptive solution when the network load is unsteady. In the following table, some of related search methods are discussed:

TABLE I. RELATED SEARCH METHODS

Search Technique	Query Forwarding	Advantages	Disadvantages
Random Walks (RW)	The queries are forwarded to only k neighbors which are selected randomly.	The complexity of produced messages is small.	Variable performance, hence success rate and number of hits changes because of random choice of neighbors for routing queries.
		Obtains local load balancing as all the nodes in the network are treated equal.	Queries for popular and unpopular objects are propagated in the same way.
Local Flooding with K Independent Random Walks (LFKIRW)	Conciliation between flooding and random walks.	It has advantages of both flooding and random walk.	More hops for traveling walkers occurs high message overhead.
		The message complexity is small, if flooding occurs locally.	
Adaptive Probabilistic Search (APS)	K independent walkers are deployed and forwarding will be done in a probabilistic manner. In order to forward the query, the neighbor is selected based on probability given in its local index.	Peer's search knowledge shared, refined and adjusted by peers.	In probabilistic selection of peers, some past attributes of peers such as bandwidth, storage and degree are not considered.
		Probabilistic selection of nodes instead of random selection.	Duplicate messages are considered as failure states.
Distributed Search Technique (DST)	The nodes in the network are classified as ordinary and power nodes. K-walkers are propagated through the network. Queries are routed through nodes in the query cache. If the query keyword t is not found in the query cache, queries are routed through both ordinary and power nodes based on past experience.	Applies Q-learning for selecting target nodes.	The messages from free riders are propagated through all the nodes.
		Improved search performance.	Required to maintain a few indices.
		Reduction in message traffic with less hop distances.	Progress of learning is the main factor of performance.
		Adaptive behaviour.	
Intelligent Search Method based on Machine Learning Techniques (ISMMLT)	In each iteration of the search, k-walkers are selected and the queries are forwarded to nodes in query caches. In order to locating the query keyword t, walkers with high p-value participate in search based on past experience.	Improved search performance	Due to selecting of value of k, overhead is high.
		Applies learning automata for training nodes in search	

K-random walks algorithm [11-13] is a kind of popular random search method that each node uses k walkers. When an intermediate node receives a random walker, it checks whether the object is available or not. If the object is not found it forwards the query to a randomly selected neighbor until TTL is terminated.

Local Flooding with k Independent Random Walks (LFKIRW) [9] starts the search with flooding the queries until discovering k neighbors which value of k is determinate before. Search is successful when one of the neighbors has the object otherwise each of k nodes begins an independent random walk.

Adaptive Probabilistic Search (APS) [8-9][14-15], relies on utilizing k independent walkers and probable forwarding. Each intermediate node, forwards the query to its neighbor with the stored probability value in its local index. Index values are updated by receiving feedback from the walkers. APS increases reliability and improves bandwidth consumption.

Distributed Search Technique (DST) [16-17], 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 Method base on Machine Learning (ISMMLT) [24] 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 ISMMLT each peer randomly selects the value of its tables in different iterations, such that all the nodes will participate in the selection process for each iteration.

III. LEARNING AUTOMATA THEORY

A learning automaton [21-22] 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 relationship between the learning automaton and the random environment is shown in fig. 1.

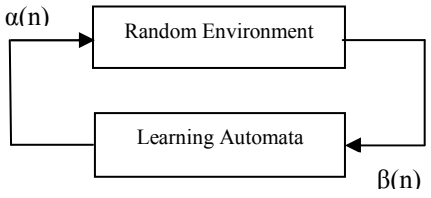

Figure 1. The relationship between the learning automaton and its random environment [22].

Learning automata can be classified into two main families [22]: 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.

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

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

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. The following describes some convergence results of the learning automata.

One form of learning automaton is KSALA [23] which k actions are selected instead of one action. The response of environment is received in different methods. In "majority of polls" method, the favorable response is when the effect of $^{k}/_{2} + 1$ selected actions in the environment have the same response, otherwise, the response is unfavorable.

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. The KSALA can be described by $\{\alpha, \beta, p, T\}$ where $\alpha \equiv \{\alpha_1, \ldots, \alpha_r\}$ denotes the set of actions, $\beta \equiv \{\beta_1, \beta_2, \ldots, \beta_m\}$ is the set of inputs, $p \equiv \{p_1, \ldots, p_1\}$ denotes the probability vector of each choice and $p(n+1) \equiv T[\alpha(n), \beta(n), p(n)]$ is the learning algorithm. $\alpha^h(s) \equiv \{\alpha_1, \ldots, \alpha_{r_s^h}\}$ denotes the set of actions for choosing h[th] action in s[th] step where $\alpha^1(s) \equiv \alpha$. In step s, the h[th] selected action is deleted from the set of all actions. Therefore it is not selected for next actions. Let α_i is the h[th] selected action in n[th] step, then the probability vector is rewarded by (3) and penalty is updated as given in (4).

$$\begin{cases} p_i^{h+1}(n) = p_i^h(n) + a\left[1 - p_i^h(n)\right] \\ p_i^{h+1}(n) = (1-a)p_j^h(n) \quad \forall j \;\; j \neq i \end{cases} \quad (3)$$

$$\begin{cases} p_i^{h+1}(n) = (1-b) + p_i^h(n) \\ p_i^{h+1}(n) = (b/r-1) + (1-b)p_{ji}^h(n) \quad \forall j \;\; j \neq i \end{cases} \quad (4)$$

According to above equations, all the members of probability vector set are updated k times.

IV. DESIGN

In this paper, the learning automaton has been used for training the nodes in the network for selection of the desired neighbor nodes.

In our distributed proposed algorithm, each set of k selected neighbors utilizes KSALA. It means that there is a table with k nodes and k probability values of locating objects based on past iterations. According to feedback from environment, all the probability values of the current node are updated.

A. How the search algorithm works?

Here, we describe the search algorithm in details. At the first step with no past iteration, for each node k neighbors are selected randomly with the probability value of *1/k*. Every time a node forwards a query to the neighbor, a new row is added to the table. If the object is not found, locating the object will continue by selecting *k* neighbors with the highest probability values among all neighbors. Selected neighbors propagate query messages in order to locate the object. After receiving the feedbacks from environment, if *k/2* of selected neighbors find the object, probability values of selected neighbors would be increased, otherwise updating the punishment of probability values is done. It is performed by KSALA algorithm. Search is terminated when the object is found or TTL is finished.

To update the probability values in order to evaluate rewards/penalties in our search algorithm we apply (1) and (2) in section III. Figure 2 and fig. 3 describe updating the probability vector and proposed search algorithm, respectively:

```
*/N ≡ Number of all the neighbors, K≡ Number of selected neighbors/*
If the feedback of [(k/2)+1] neighbors is "HIT"  //majority of polls
    The neighbors with "Hit" feedback updated by EQ (3)
    and
    the others updated by by EQ (4)      //the neighbors with "MISS"
else
    all k selected neighbors updated by EQ (4)
```

Figure 2. Updating probability vector according to feedbacks

```
*/ CN ≡ Current node for search, Q ≡ Query Keyword
K ≡ The number of selected neighbors, P_Vector ≡ The set of probability values/*
1. User submits a query
2. Search query node for Q
3. If Q is not in CN
  3.1. Select K neighbors with the highest P_vector among all neighbors
       Generate K query messages
       Search starts with k neighbors
4. If a hit occurs, sent back results on the reverse path
5. All nodes on the path update appropriate P_vectors with KSALA
```

Figure 3. Proposed search algorithm

V. SIMULATION

In this section, hypotheses of simulation are stated first and result of the simulation are investigated then.

Some network criteria such as overload due to the messages generated in the network are compared in addition to search quality criteria like success of the search and average number of the objects found per one query with the K-random walks algorithm, LFKIRW and APS method.

A. Hypotheses of Simulation

We use OverSim [26] to simulate our search algorithm and evaluate our experiments.

First we describe our network model and then set the required parameters of simulation. We construct our network using a random graph [11] with 1000 nodes for different simulations. The average out-degree of each node is 3. There are 100 distinct objects distributed in various nodes randomly. The maximum time to deliver the messages of the queries is called TTL and is assigned 6. The node failure rate is 20% in our simulations. The maximum selected walker is 15. A standard learning automaton is used with L_{RP} algorithm, in which the initial values of a and b are both considered 0.5. Table II shows the summary of the simulation parameters with the default values.

TABLE II. SIMULATION PARAMETERS

Parameters	Default values
Network Topology	Random Graph
Network Type	Unstructured
No. of Objects	100
No. of Maximum Walkers	15
Time To Live	6

B. Evaluation of the Proposed Algorithm

Performance of the proposed search algorithm is evaluated by implementing different simulations. The method developed here is compared with K-random walks algorithm, LFKIRW and APS method. The following graphs show our results of simulations with regard that k is the number of the walkers in different algorithms. In our simulations, walkers vary from 1 to 15.

In fig. 4, it is shown that the success rate of k-random walks algorithm due to random selection of walkers with no feedback from environment is low but in our proposed algorithm after the 5th selection of walker- in this case, the probability values of neighbors are updated sufficiently -it is growing up to 85%. Selecting neighbors with high priority based on past iterations is the main reason for high success rate for our algorithm. The average success rate of search for our algorithm is about 65% while LFKIRW and APS are respectively 53% and 60%.

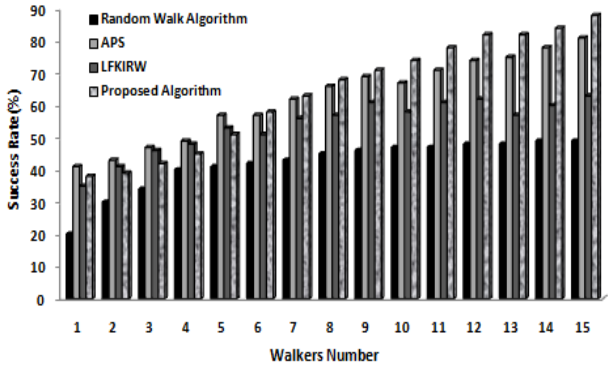

Figure 4. Success rate vs. number of deployed walkers

The number of discovered objects per query is presented in fig. 5. Our proposed algorithm has more precise results than LFKIRW and k-random walks and it is close to APS. Because in our algorithm the selected walkers always have the highest probability value, it will increase the chance of discovering objects when the query is propagated.

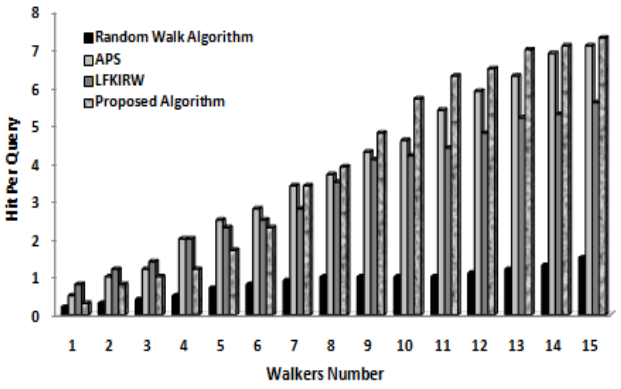

Figure 5. Number of hits per query vs. number of deployed walkers

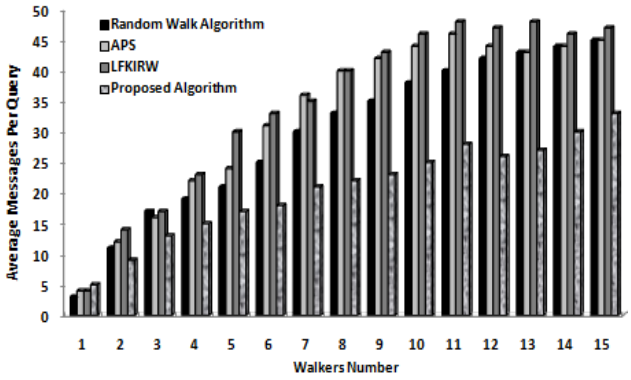

Figure 6. Message per query vs. number of deployed walkers

Figure 6 gives a comparison between the proposed algorithm and the other algorithms taking into account the overload generated from messages of each query. It can be observed in this diagram that the proposed algorithm provides a much smaller average number of messages generated for each query, because no additional message and query is sent for this purpose into the network. The obtained overload from generation of the messages is very significant in K-random walk method since the queries are randomly sent.

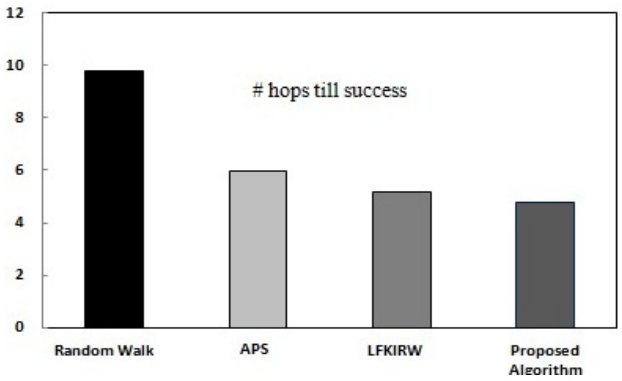

Figure 7. Delay comparison

Delay of search is addressed based on the average number of hops visited for a single search. Less hops means smaller delay and greater speed for discovering the objects. By looking at fig. 7, one may notice that the proposed algorithm has a smaller delay in comparison with the other methods, which can be attributed to the intelligent selection of the walkers during the search in the developed algorithm.

VI. CONCLUSION

An adaptive version of k-random walks algorithm for peer-to-peer networks utilizing learning automata was introduced in this paper. Our proposed algorithm is based on adaptive select of walkers according to each node's feedback. The suggested search algorithm considers the feedbacks from the environment and then makes decision about suitable nodes in order to participate in search. By applying LA for each set of k nodes in the network, all the neighbors with the highest probability value of successful search in the past iterations are selected adaptively to continue the search. We compared the performance of our algorithm with k-random walks algorithm, APS and LFKIRW via different simulations. The simulation results show that our algorithm can effectively improve the success rate, the number of discovered objects and the generated messages per query in comparison with other methods and has less delay than k- random walks and APS and LFKIRW.

REFERENCES

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

[2] S. Ratnasamy, P. Francis, M. Handley, and R. Karp, "A scalable content-addressable network," Proc. Of SIGCOMM, 2001.

[3] R. Karger, D. Kaashoek, and M. Balakrishnan, "Chord: A scalable peer-to-peer look up service for internet applicatons," Proc. Of SIGCOMM, 2001.

[4] P. Rowston, and M. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," Proc. Of IFIP / ACM Middleware, Heidelberg, Germany, 2001.

[5] G.H. Page.[online], Available: http://www.gnutella.wego.com.

[6] L. Hoffman, "Napster and other Internet peer-to-peer applications," George Washengton University, available: citeseer.ist.psu.edu/kim01pricing.html, 2002.

[7] 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.

[8] X. Li, and J. Wu, "Searching techniques in peer-to-peer networks," Handbook of Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks, CRC Press, Boca Raton, FL, 2005.

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

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

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

[12] G. H. Fletcher, H. A. Sheth and K. Borner, "Unstructured peer-to-peer networks: topological properties and search performance," Lecture Notes in Computer Science- Agent and Peer-to-Peer Computing, Springer Berlin/ Heidelberg, 2005, pp. 14-27.

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

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

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

[16] M. Ghorbani, M. R. Meybodi, and A. M. Saghiri, "A novel self-adaptive search algorithm for unstructured peer-to-peer networks utilizing learning automata," Proc. Of the 3rd Joint Conf. Robotics & AI and the 5th Robocup Iranopen Int. Symposium, Qazvin, Iran, April 8, 2013, pp. 42-47.

[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, 2013 ,pp. 145-149.

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

[19] 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 January 2008, pp. 128-135.

[20] R. S. Sutton and A. G. Barto, "Reinforcement learning: introduction," in Proceeding of the MIT Pres, 1998.

[21] E. Mance, and S. H. Stephanie, "Reinforcement learning: A tutorial," Proc. Of the Wright Laboratory, 1996.

[22] K. Najim, and A. S. Poznyak, "Learning automata: theory and application," Proc. Of the Tarrytown, New York, Elsevier Science Publishing Ltd., 1994.

[23] S. M. Abolhasani, and M. R. Meybodi, "LADIT: Learning automata based protocol for routing in sensor networks," 2th Conference on Sensor Networks, Yazd, Iran, 2008, pp. 20-33.

[24] F. Torabmostaedi, and M. R. Meybodi, "An intelligent search method based on machine learning for peer-to-peer networks," Int. Conf. Contemporary Issues in Computer and Information Sciences, Zanjan, 2011, pp. 495-500.

[25] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti, "A local search mechanism for peer-to-peer networks," 11th Conf. Information and knowledge management, 2002, pp. 300-307.

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