A New Recommendation Algorithm Using Distributed Learning Automata and Graph Partitioning

Shahrzad Motamedi Mehr Majid Taran
Department of Electrical and Computer Engineering
Islamic Azad University, Qazvin Branch
Qazvin, Iran
motamedi@tmu.ac.ir, m_taran@isc.iranet.net

Ali B. Hashemi M. R. Meybodi
Computer Engineering and Information Technology
Department, Amirkabir University of Technology
Tehran, Iran
{a_hashemi, mmeybodi}@aut.ac.ir

Abstract— **Recommendation systems aim at directing users toward the resources that best meet their needs and interests. In this paper, we propose a new recommendation algorithm based on a hybrid method of distributed learning automata and graph partitioning. The proposed method utilizes usage data and hyperlink graph of the web site. The idea of the proposed method is that an appropriate recommendation for a user can be pages similar to the pages the user has already visited. To calculate similarities between pages, it is assumed that if different users request a couple of pages together, these pages are likely to correspond to the same information need therefore can be considered similar. Experiments on synthetic and real data show that the proposed algorithm provides better recommendations than the only learning automata based recommendation method reported in the literature.**

Keywords-web usage mining, distributed learning automata, web recommendation

I. INTRODUCTION

Determining similarities between web pages is one of the goals of web mining methods. A simple approach, yet expensive and manual, to address this problem is using the knowledge of some highly trained experts to classify the web pages into many subjects. Another approach is assigning keywords to each web page, which creates new problems like tackling with homonymous and synonyms and finding new keywords for new documents. Although keyword can be extracted automatically form a text document, this task is more difficult for the multimedia documents. An old approach to find some similarities between documents is mining on citations in scientific papers, which eliminate most of the mentioned problems.

Heylighen and Bollen [1] introduced a new approach to this problem using usage data. This approach is based on the idea that if two documents respond to the same information need, then these documents are similar. This method assumes that users approximately know about the pages they are going to visit and they do not move to another page randomly. By moving among the pages, users create a virtual relation between pages, which is based on their mental model and are not necessarily like the existing relations like links between web page and keywords defined for articles. This virtual relation can be used to represent similarity between pages since it is assumed that users have enough information about the web pages they visit. In this approach neither there is any need to have experts to determine keywords or to classify the pages, nor it requires explicit users interactions, like rating to an article. In this approach similarities between web pages are evolved according to the rule of Hebb[2] such that the more users move from page *i* to page *j*, the more similar these pages will be considered. In extensions to this algorithms, the transitive relation between two pages is considered such that any page *k* which are visited after page *j* is also considered to be similar to page *i* with a decreasing coefficient proportional to the distance to page *i*. Hashemi and Meybodi [4] introduced another algorithm based on distributed learning automata in which, like Hebbian algorithm, if two web pages of a website are visited consecutively they are considered to be similar and reported more accurate similarity compared to both Hebbian algorithm and DLA-based algorithm proposed in [3].

Forsati et al. introduced web page personalization algorithm based on distributed learning automata in which learn the similarity between documents using the usage data and behavior of previous users[5].

Learning automata and distributed learning automata have been used in many other web mining applications. Anari et al. [6] used learning automata to determine the weight of hyperlinks between web pages in order to compute rank of each web pages. Forsati and Meybodi [7] introduced a web page recommendation algorithm based on distributed learning automata in which DLA learn the behavior of previous users' and recommend pages based on the learned patterns.

978-1-4577-2150-2

In [8], we introduced a new hybrid method of distributed learning automata and graph partitioning to determine similarity between web pages using the web usage data.

Numerous approaches are introduced for recommendation system which can be categorized into two major groups, which are content-based filtering agents and collaborative filtering systems [9].

Some studies have considered the approach of using pages interesting to the user for the recommendation process. In [12], Mobasher et al. use statistical significance testing to judge whether a page is interesting to a user. Its main idea is: A duration threshold is calculated for each page using the average duration and standard deviation of the visits to the page; if the duration of a page is longer than the threshold, that page is considered interesting to the user and vice versa.

In [10, 11], Nakagawa and Mobasher use association rule mining, sequential pattern mining, and contiguous sequential mining to generate three types of navigational patterns in the off-line phase. In the on-line phase, recommendation sets are selected from the different navigational models, based on a localized degree of hyperlink connectivity with respect to a user's current location within the site.

In this paper, we propose a new recommendation algorithm based on a hybrid method of distributed learning automata and graph partitioning. The proposed algorithm utilizes the web usage data and the hyperlink graph of the web site to determine the similarities between web pages then recommends the user web pages which the user is more likely to visit. Empirical results on a synthetic model of a web site and its users [8] show that the proposed algorithm can determine similarities between web pages better than a distributed learning automata based methods introduced in literature[4] as well as Hebbian and extended Hebbian algorithms [1]. In addition, it is shown that for a real web site, the proposed algorithm provides better recommendations than the only learning automata based recommendation method reported in the literature[7].

The rest of this paper is organized as follows. Section 2 introduces learning automata and distributed learning automata. Section 3 *PageRank* algorithm respectively. The proposed algorithm is introduced in section 4. Section 5 gives the results of experiments. The last section is the conclusion.

II. LEARNING AUTOMATA

Learning Automata are adaptive decision-making devices operating on unknown random environments. The automata approach to learning involves the determination of an optimal action from a set of allowable actions. An automaton can be regarded as an abstract object which has finite number of possible actions. In each decision process, the automata select an action from its finite set of actions. This action is applied to a random environment. The random environment evaluates the selected action and provides a response to the applied action of automata. The response of the environment is used by the automata to learn the optimal action [13][14][22].

A. Distributed Learning Automata

Distributed learning automata [15] 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 [23][27].

III. PAGERANK ALGORITHM

The PageRank algorithm for ranking hypertext-linked web pages was introduced in [16] and [17]. The idea of the PageRank algorithm is that a link from page v to page u is an evidence of the importance of page v. In particular, the amount of importance conferred on page v by page u is proportional to the importance of page u and inversely proportional to the number of links in page u. Since the importance of u is itself not known, determining the importance for every page requires an iterative fixed-point computation. To do so, the PageRank algorithm can be thought of as a model of the behavior of a random surfer[16], who either selects an outgoing link from the page currently visiting or jumps to a random page. In this model, the PageRank of a web page i is defined as the probability of the random surfer being at page i and can be calculated as follows.

Let $u \rightarrow v$ denote the existence of a link from page u to page v in the hyperlink graph of web, G. Let $deg(u)$ be the out-degree of page u in G, i.e. the number of links to different web pages in page u. The probability of the random surfer, currently visiting page i, lands at a the page j, $v_j \in \{v \mid u_i \rightarrow v\}$ equals to p_{ij} (Eq. 1).

$$p_{ij} = \begin{cases} \dfrac{1}{\deg(u_i)} : if \quad (u_i \rightarrow u_j) \in E \\ 0 \qquad : otherwise \end{cases} \tag{1}$$

The PageRank of a page i is defined as the probability that at some particular time step $k>K$, the surfer is at page i. Consider the Markov chain induced by the random walk on G, where the states are given by the nodes in G, and the stochastic transition matrix describing the transition from i to j is given by matrix P where p_{ij} is defined by Eq. 1. For P to be a valid transition probability matrix every node must have at least one outgoing link; i.e. P should have no rows consisting of all zeros. To solve this problem a new transition matrix P' is defined where all states have at least one outgoing transition in the following way.

Let n be the number of nodes (pages) in the Web graph. Let $\vec{v}$ be the n-dimensional column vector representing the probability distribution of destination pages when a random jump is made. Typically, this distribution is a uniform probability distribution over all nodes. Let $\vec{d}$ be the n-dimensional column vector where $d_i=1$ if page i has no outgoing link and $d_i=0$ otherwise. Then, matrix P' is constructed as Eq 2.

$$D = \vec{d}\,\vec{v}^{T}$$
$$P' = P + D \tag{2}$$

In order to have a unique stationary probability distribution for the Markov chain defined by P', P' should be aperiodic and irreducible; the former holds for the Markov chain induced by the Web graph. The latter holds if G is strongly connected, which is usually not the case for the web graph. In the page rank to ensure this property a complete transition graph with small transition probabilities is created by adding a teleportation matrix E to P' as Eq. 3.

$$E = [1]_{n\times 1}\,\vec{v}^{T}$$
$$P'' = cP' + (1-c)E \qquad ,0 \le c \le 1 \tag{3}$$

In terms of the random walk, the effect of teleportation matrix E is as follows. At each time step, with probability $(1-c)$, a surfer visiting any page i will jump to a random page rather than following one the links in the page i. The destination of the random jump is chosen according to the probability distribution given in $\vec{v}$.

By redefining the vector $\vec{v}$ given to be non-uniform, so that random jumps can be non-uniform, the resulting PageRank vector can be biased to prefer certain kinds of pages. For this reason, vector $\vec{v}$ is called the personalization vector [18].

In [19], Eirinaki and Vazirgiannis proposed a usage-based personalized PageRank (UPR) algorithm used for ranking the Web pages of a site based on the navigational behavior of previous users. Let w_i be the weight of page i which represents the number of times page i was visited, and the w_{ij} be the weight of transition from page i to page j which represents the number of times page j was visited immediately after page i. In UPR, the transition probability of page i to page j is proportional to the usage of the link to page j in page i and is calculated as Eq. 4 .

$$p_{ij}^{UPR} = \frac{w_{ij}}{\sum_{x_k \in out(k)} w_{ik}} \tag{4}$$

where $out(i)$ denotes the set of pages linked from page i. The personalization vector $\vec{v}$ is defined as Eq. 5 .

$$v_{ij}^{UPR} = \frac{w_i}{\sum_{\forall k} w_k} \tag{5}$$

IV. The Proposed Algorithm

The proposed algorithm recommends new web pages to a user based on the navigation history of the user, the usage data and the hyperlink graph of the web site. To do so, the hyperlink graph of a web site is partitioned by a graph partitioning algorithm. Then a distributed learning automaton is used to determine similarities of the web pages based on the assumption that two consecutive visited pages are similar if they have been partitioned into the same partition [8]. Then a new usage-based page rank (UPR-DLA) is calculated using the determined similarity between web pages. Afterwards, the calculated page rank and the similarity between web pages are used to build a Markov model of the users' transitions in the web site. Finally, this Markov model is used to predict the probability of visiting new pages based on which web pages are recommended to the users.

A. Determining Similarity between Web Pages

In [8], we proposed a new algorithm for determining the similarity between each pair of pages using the user interactions and hyperlink graph of a web site using distributed learning automata and graph partitioning.

The amount of similarity between documents is calculated when the user surf from one document to another. This feature causes the algorithm is used online. The proposed algorithm does not use any information about the content of documents and only using user behavior will calculate the amount of similar documents with each other.

In the proposed algorithm, the hyperlink graph of a web site is partitioned by a graph partitioning algorithm. Then a distributed learning automaton is used to determine similarities of the web pages based on the assumption that two consecutive visited pages are similar if they have been partitioned into the same partition. We use this method to determine similarity between web pages.

B. Usage-based Pagerank based on DLA

In order to calculate the page rank of the web pages, the usage-based page rank algorithm [7] is applied. In this method, the transition probability of page i to page j is set to the probability of action j in learning automaton i (Eq. 6) and personalization vector $\vec{v}$ is calculated according to Eq. 5.

$$p_{ij}^{DLA-UPR} = s(i,j) \tag{6}$$

In this algorithm, the transition matrix P and personalization vector $\vec{v}$ in the original PageRank algorithm are computed based on usage data instead of link structure. For this reason, a DLA learns the transition probability matrix P from the behavior of the visiting users. In addition, the personalization vector $\vec{v}$ is computed based on the visiting rate of pages preferring pages which are visited by more users. Having the transition matrix P and personalization vector $\vec{v}$ which are acquired from previous users' visits, the PageRank algorithm is used to compute the rank of each page Eq. 7.

$$DLA-UPR(p_i) = d.\sum_{p_j \in in(p_i)} \frac{DLA-UPR(p_j)}{|out(p_j)|} + (1-d)\frac{1}{N}$$
$$DLA-UPR(p_i) = d.\sum_{p_j \in in(p_i)} s(j,i) + (1-d)\frac{1}{N} \tag{7}$$

where $out(p_j)$ denotes the set of pages linked from page j and $in(p_j)$ is the set of pages which have a link to page j.

$p_1, p_2, ..., p_N$ are the pages under consideration, N is the total number of pages and d is an adjustment coefficient between zero and one. For the proposed algorithm, d is set to 0.85.

As an improvement of the PageRank algorithm, the PageRate algorithm also models a *random surfer's navigation* on the link structure of a Web site consisting of N nodes. A surfer is given a Web page at random. The surfer has two choices on each Web page.

First, the surfer clicks each hyperlink on the page with a probability, which is the number of user traversals on the hyperlink divided by the sum of user traversals on all the out-links of the page multiplied by the damping factor d . Second, the surfer jumps to any page on the Web site link structure with an equal probability $(1- d) / N$. The surfer keeps clicking hyperlinks on the following pages. The PageRate of a page A is calculated as Eq. 8 .

$$PR(A) = \frac{1-d}{N} + d \times \left\{ PR(T_1) \times \frac{n(T_1, A)}{\sum_n n(T_1, n)} + ... + PR(T_n) \times \frac{n(T_1, A)}{\sum_i n(T_n, i)} \right\} \quad (8)$$

where $n(T_j,A)$ is the number of user traversals on the hyperlink from page T_j to page A. $\sum_i n(T_j, i)$ is the sum of user traversals on all the hyperlinks of page T_j.

In addition, we propose a PageRate method to give web pages on a web site ratings based on the Web link structure and usage data. The proposed PageRate method has two phases. First, the Web link structure and the Web users' visiting behaviors are used for page rating. Second, Web pages are compared with each other on the similarity of incoming links. the PageRate algorithm is used to compute the rank of each page Eq. 8.

C. Page Recommendation

The goal of a recommendation system is to produce a set of pages for a user, which are not visited by them and are most suited with the user's interests [19]. Suppose that a user is walking in the web site and the path traversed by him is $p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_k$. We use a fixed-size sliding window over the user's active session to capture his path. Note that the sliding window of size $|w|$ over the active session allows only the last w visited pages to influence the recommendation set.

To recommend page p_{k+1} to the current user, we use Markov models to model the navigational behavior of the users of a web site. In the proposed model, the order of the Markov model indicates the "memory" of the prediction, i.e. denotes the number of previous user steps which are taken into consideration in the process of calculating the path probabilities. Note that, the selection of the order influences both the prediction accuracy and the complexity of the model while heavily depends on the application. In the proposed recommendation algorithm, after computing the transition matrix P using DLA, the path probabilities are computed for an m-order model using the chain rule as follows:

$$\Pr(p_1 \rightarrow p_2 \rightarrow p_3 \cdots \rightarrow p_k) = \Pr(p_1) \times \prod_{i=2}^{k} \Pr(p_i \mid p_{i-m} \cdots p_{i-1}) \quad (9)$$

For example the probability of path $p_1 \rightarrow p_2 \rightarrow p_3$ equals to:

$$\Pr(p_1 \rightarrow p_2 \rightarrow p_3) = \Pr(p_1) \Pr(p_2 \mid p_1) \Pr(p_3 \mid p_2) = \Pr(p_1) \frac{\Pr(p_1 \rightarrow p_2) \Pr(p_2 \rightarrow p_3)}{\Pr(p_1) \Pr(p_2)} \quad (10)$$

where $\Pr(\bullet \rightarrow \bullet)$ represents the probability of transition between pages and $\Pr(\bullet)$ is the rank of page obtained from the biased PageRank and PageRate algorithm.

Based on Eq. (9), the prediction of the next most probable page which will be visited by the user is performed by computing the probabilities of all existing paths such $p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_k \rightarrow p_{k+1}$ having the pages visited so far by the user as prefix and recommending the user the most probable web page. The bounded probabilities' computation is straight-forward since it reduces to a lookup on the transition probability matrix P. For all unvisited pages, their visiting probability is computed. Then, the D most probable web pages among them are selected to be recommended to the user. The seudocode of the proposed algorithm is depicted in Figure 1.

Procedure *Proposed Recommendation Algorithm*
n: number of pages in the web site
UPR($page_i$): Usage-based Page Rank of page i
s(i,j) : similarity between page i and j
$history_u$(t): Pages user u has already visited at time t.
$page^u$(t): t^{th} page which user u has visited in the web site.
D: Number of recommendation pages
begin
  $p(page_i)$ = DLA-UPR($page_i$) i=1, 2, ..., N
  $p(page_i \rightarrow page_j) = s(i,j)$ i,j=1, 2, ..., N
  for *each $page_i \notin history_u$* ***do***
    Calculate recommendation values for $page_i$:
    $r_i^u(t) = Pr(page^u(t\text{-}w) \rightarrow page^u(t\text{-}w+1) \rightarrow ... \rightarrow page^u(t) \rightarrow page_i)$
  end-for
  Create the recommendation list with D pages with the most recommendation value $r^u(t)$.
end

Figure 1. Pseudo code of the proposed recommendation algorithm

V. EXPERIMENTATIONS

In this section we explain the model used for generating web usage data then presents experimental results of the proposed algorithm and compare the result with previous algorithms in the literature.

A. Web Site and Users Model

We uses the web access logs of the DePaul University CTI Web server [20], based on a random sample of users visiting the site for a 2 week period during April 2002. This dataset contains 13745 distinct user sessions of length more than 1 and 683 distinct pages. We split the data set in to training and test data set using k-fold cross-validation. These two parts are non-overlapping. The advantage of this method over repeated random sub-sampling is that all observations are used for both

training and validation, and each observation is used for validation exactly once [25].

B. Performance measure

In order to evaluate performance of the proposed recommendation algorithm, we use two well-known measures, namely recommendation precision and coverage [21][24][26] which are similar to precision and recall known from information retrieval, respectively. Recommendation precision shows the exactness of the recommendation list generated by an algorithm. It is calculated as the ratio of the number of correct recommendations to the number of recommendations. A correct recommendation for a session s is a recommended page that is visited in the rest of session s. If the algorithm uses the p first pages of a session s to generate a recommendation list $R(p)$ for the rest of the session $T(p)$, the recommendation precision is calculated as the Eq. 11.

$$\text{Precision} = \frac{|R(p) \cap T(p)|}{|R(p)|} \tag{11}$$

Recommendation coverage shows completeness of the recommendation list generated by an algorithm. It is the ratio of the number of correct recommendations a session s to the number of remaining pages in that session $T(p)$ (Eq. 12).

$$\text{Coverage} = \frac{|R(p) \cap T(p)|}{|T(p)|} \tag{12}$$

Ideally, one would like high precision and high coverage. A single measure that captures this is the F1 measure: (Eq. 13).

$$F1 = \frac{2 \times Coverage \times \text{Precision}}{Coverage + \text{Precision}} \tag{13}$$

The F1 measure attains its maximum value when both precision and coverage are maximized.

C. Experimental Setting

In the experiments, an implementation of multilevel graph partitioning method [28] is used to create partitions of K=25 web pages from the graph of the web site. Also, for the proposed algorithm, a_0 and b_0^{cycle} are set to 0.002 and ω and b_0 are set to 0.02 and 0.3 respectively.

D. Impact of Active Window Size on User Navigation Trail

In all experiments we measured both Precision and Coverage of recommendations against varying umber of recommended pages. In our state definition, we used the notion of N-Grams by putting a sliding window on user navigation paths. The implication of using a sliding window of size w is that we base the prediction of user future visits on his w past visits. The choice of this sliding window size can affect the system in several ways. To consider the impact of window size (the portion of user histories used to produce recommendations) on the DLA Personalization algorithm, we also vary window sizes $|w|$ from 2 to 4.

We used w+d such as minimum of length of session. The session of user path delete if length of session less than the number of recommended of pages(d).

E. Experimental Results

In this statistical method similarity between pages i and j is calculated as the number of visits of page j just after page i, divided by the number of visits of pages i (Eq. 14).

$$simpleSimilarity(i,j) = \frac{visited(i,j)}{\sum_{k=1}^{n} visited(i,k)} \tag{14}$$

Figure 2 to 9 have shown the comparison of proposed algorithm performance using PageRank and PageRate with DLA and AR method in the sense of their precision and coverage in different number of recommended pages on CTI dataset.

Figure 2 and Figure 3 show the impact of window size on precision and coverage of the Recommender algorithm. The results show clearly that precision and coverage increase as a larger portion of user's history is used to generate recommendations.

The best results when using a window of size 4 and the other hand the experiments we used a fixed window size of 4 on recommendation history (set the window size to 4).

Figures 4 and 5 show the transitive relation, assign penalty and graph partitioning to improve precision and coverage are used.

In figure 6 is shown precision of recommender algorithms compare to DLA[5][7] and AR[12]. The precision of proposed methods are more than other methods based on distributed learning automata and association rule.

Fig. 7 shows the coverage of recommender algorithms. As shown in figure 7, proposed algorithms perform better than three previously introduced DLA[5][7] and AR[12] methods.

As shown in figure 6 and 7, when we increase the number of recommendation page, the precision decreases and coverage increases.

As shown in figure 8 Performance of proposed algorithms compare to DLA[5][7] and AR[12] algorithm is better.

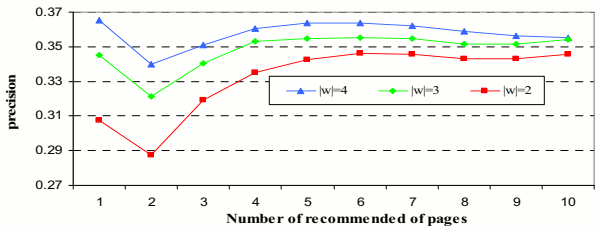

Figure 2. Proposed recommender algorithm PageRank precision with various user active windows size and w+d

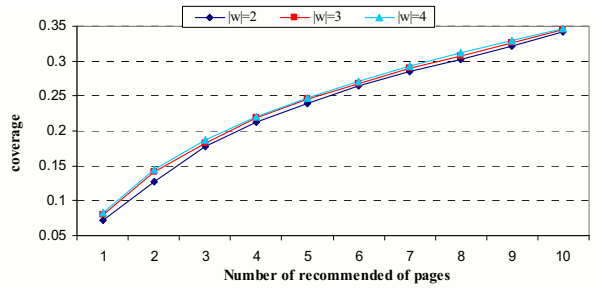

Figure 3. Proposed recommender algorithm PageRank coverage with various user active windows size and w+d

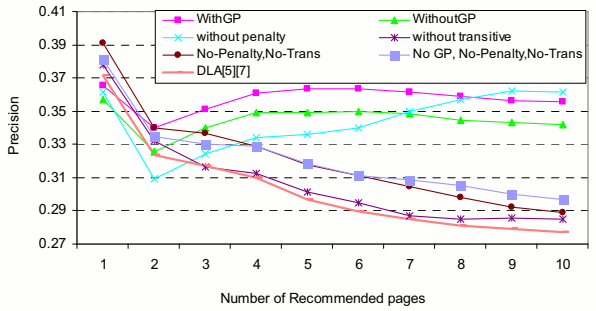

Figure 4. Proposed recommender algorithm PageRank precision with different conditions and |w| = 4 and w+d

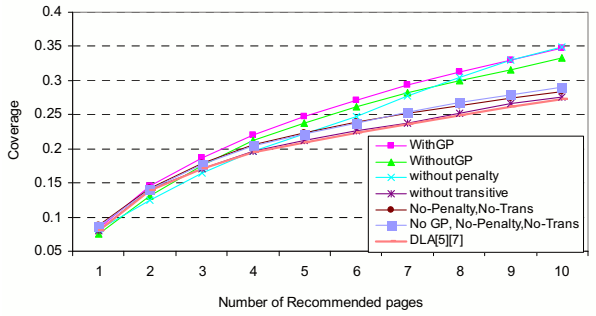

Figure 5. Proposed recommender algorithm PageRank coverage with different conditions and |w| = 4 and w+d

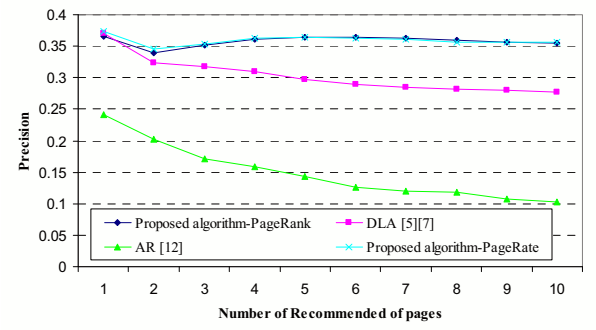

Figure 6. Precision of proposed algorithm (PageRank and PageRate) and DLA [5][7] and AR[12] methods with |w| = 4 and w+d

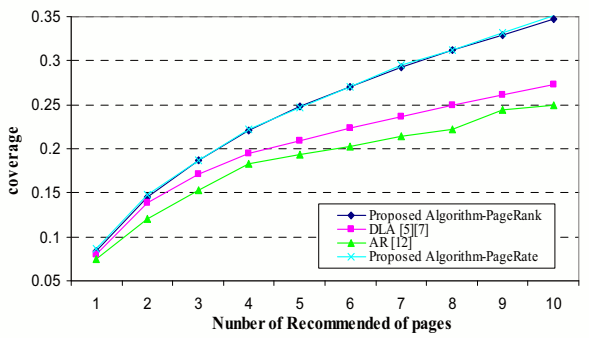

Figure 7. Coverage of proposed algorithm (PageRank and PageRate) and DLA[5][7] and AR[12] methods with |w| = 4 and w+d

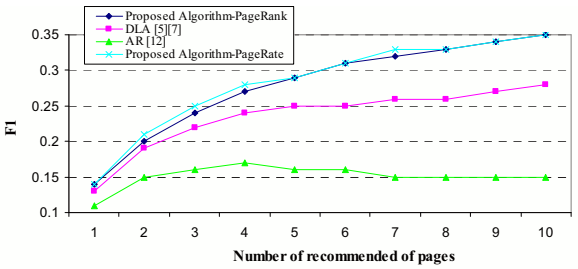

Figure 8. Performance of proposed algorithm (PageRank and PageRate) compare to DLA[5][7] and AR[12] methods with |w| = 4 and w+d

As shown in figure 9, Precision vs Coverage of proposed algorithms compare to DLA[5][7] and AR[12] algorithm are better.

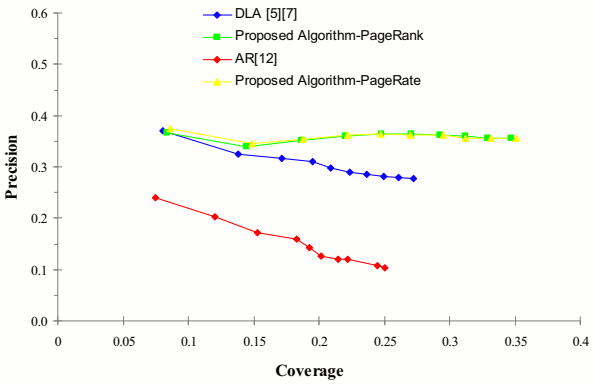

Figure 9. Precision vs Coverage of proposed algorithm(PageRank and PageRate) compare to DLA[5][7] and AR[12] methods with |w| = 4 and w+d

In the PageRate method, the more past users have followed a certain link, the more important the page is regarded.

VI. CONCLUSION

In the proposed algorithm, a distributed learning automaton tries to determine similarities between pages of a web site considering that each visited page is probably similar to all other pages visited afterwards. In addition, with the assumption that the hyperlink graph of a web site are usually such that similar pages are divided into the same partition after partitioning the hyperlink graph of the web site, partitioning

the hyperlink graph of the web site is used to help the distributed learning automata to learn similarities between web page more accurately. The pages in a recommendation list are ranked according to their importance and similarities, which is in turn computed based on proposed algorithm and *PageRank* and *PageRate* algorithm. Experimental results illustrate that the proposed algorithm can generate higher quality recommendations than using either the distributed learning automata recommendation and the association rule mining recommendation algorithm [5][7][12].

REFERENCES

[1] F. Heylighen, and J. Bollen, "Hebbian Algorithms for a Digital Library Recommendation System," IEEE Computer Society, City, 2002.

[2] D. O. Hebb, "The Organization of Behavior: A Neuropsychological Theory," Wiley-Interscience, New York, 1949.

[3] S. Saati, and M. R. Meybodi, "A Self Organizing Model for Document Structure Using Distributed Learning Automata," City, 2005.

[4] A. B. Hashemi, and M. R. Meybodi, "Web Usage Mining using Distributed Learning Automata," In Proceedings of the 12th International CSI Computer Conference (CSICC'07), Tehran, Iran, 2007.

[5] R. Forsati, M. R. Meybodi, and M. Mahdavi, "Web Page Personalization based on Distributed Learning Automata," In Proceedings of the 3rd Information and Knowledge Technology, Mashhad, Iran, 2007.

[6] Z. Anari, M. R. Meybodi, and B. Anari, "Web page ranking based on fuzzy and learning automata," In Proceedings of the International Conference on Management of Emergent Digital EcoSystems, France, 2009.

[7] R. Forsati, and M. R. Meybodi, "Effective page recommendation algorithms based on distributed learning automata and weighted association rules," Expert Systems with Applications: An International Journal, vol. 37, no. 2, 2010, pp. 1316-1330.

[8] SH. Motamedi Mehr, M. Taran, A. B. Hashemi, and M. R. Meybodi, "Determining Web Pages Similarity Using Distributed Learning Automata and Graph Partitioning," International Symposium on Artificial Intelligence and Signal Processing (AISP) IEEE Iran Section, Tehran, Iran , 2011.

[9] B. Mobasher, R. Cooley, and J. Srivastava, "Automatic personalization based on Web usage mining," Communications of the ACM, vol. 43, no. 8, 2000, pp. 142-151.

[10] M. Nakagawa, and B. Mobasher, "A hybrid web personalization model based on site connectivity," In Proceedings of the 2003 WebKDD Workshop Held at KDD'2003, Washington, DC, USA, 2003.

[11] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, "Improving the effectiveness of collaborative filtering on anonymous web usage data," In Proceedings of the Workshop on Intelligent Techniques for Web Personalization (ITWP01), Seattle, USA, 2001.

[12] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, "Effective personalization based on association rule discovery from web usage data," In Proceedings of the 3rd international workshop on Web information and data management, Atlanta, Georgia, USA, 2001.

[13] K. S. Narendra, and M. A. L. Thathachar, "Learning automata: An introduction," Prentice Hall, 1989.

[14] M. A. L. Thathachar, and R. H. Bhaskar, "Learning automata with changing number of actions," IEEE Transactions on system, man and cybernetics, vol. 17, no. 6, 1987, pp. 1095-1100.

[15] H. Beigy, and M. R. Meybodi, "Utilizing Distributed Learning Automata to Solve Stochastic Shortest Path Problem," International Journal of Uncertainty, Fuzziness and Knowledge-based Systems, vol. 14, no. 5, 2006, pp. 591-617.

[16] S. Brin, and L. Page, "The anatomy of a large-scale hypertextual Web search engine," Computer Networks and ISDN Systems, vol. 30, no. 1-7, 1998, pp. 107-117.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: bringing order to the web," Computer Science Department, Stanford University, Palo Alto, California, USA, 1999.

[18] S. Kamvar, T. Haveliwala, and G. Golub, "Adaptive methods for the computation of PageRank," Linear Algebra and its Applications, 386(2004), pp. 51-65.

[19] M. Eirinaki, and M. Vazirgiannis, "Web site personalization based on link analysis and navigational patterns," ACM Transactions on Internet Technology, vol. 7, no. 4, 2007, pp. 21.

[20] CTI DePaul web server data http://maya.cs.depaul.edu/classes/ect584/data/cti-data.zip.

[21] C. -N. Ziegler, G. Lausen, and L. Schmidt-Thieme, "Taxonomy-driven computation of product recommendations," In Proceedings of the Thirteenth ACM international conference on Information and knowledge management, Washington, D.C., USA, 2004.

[22] M. Thathachar, and P. Sastry, "Varieties of learning automata: an overview," IEEE Transactions on Systems, Man and Cybernetics, vol. 32, 2002, pp. 711-722.

[23] H. Beigy, and M. R. Meybodi, "Utilizing Distributed Learning Automata to Solve Stochastic Shortest Path Problem," International Journal of Uncertainty, Fuzziness and Knowledge-based Systems, vol. 14, no. 5, pp. 591-617, 2006.

[24] J. Li, and O. R. Zaïane, "Using distinctive information channels for a mission-based Web recommender system," In Proceedings of the WebKDD- 2004 workshop on Web Mining and Web Usage Analysis, part of the ACM KDD: Knowledge Discovery and Data Mining Conference, Seattle, WA., 2004.

[25] G. J. McLachlan, and K. A. Do, and C. Ambroise, "Analyzing microarray gene expression data," Wiley, 2004.

[26] J. Li, and O. R. Zaïane, "Combining Usage, Content, and Structure Data to Improve Web Site Recommendation," In E-Commerce and Web Technologies, Lecture Notes in Computer Science, vol. 3182, 2004, pp. 305-315.

[27] H. Beigy, M. R. Meybodi, "A New Distributed Learning Automata Based Algorithm For Solving Stochastic Shortest Path Problem," Proceedings of the Sixth International Joint Conference on Information Science, Durham, USA, 2002, pp. 339-343.

[28] G. Karypis, and V. Kumar, "METIS: A Software Package for Partitioning Unstructured Graphs," Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Minneapolis, City, 2007.