A Novel Time Series Link Prediction Method: Learning Automata Approach

Behnaz Moradabadi and Mohammad Reza Meybodi
Department of Computer Engineering
Amirkabir University of Technology
Tehran, Iran
moradabadi@aut.ac.ir
mmeybodi@aut.ac.ir

Abstract. The ability to predict linkages among data objects is central to many data mining tasks, such as product recommendation and social network analysis. Substantial literature has been devoted to the link prediction problem either as an implicitly embedded problem in specific applications or as a generic data mining task. This literature has mostly adopted a static graph representation where a snapshot of the network is analyzed to predict hidden or future links. However, this representation is only appropriate to investigate whether a certain link will ever occur and does not apply to many applications for which the prediction of the repeated link occurrences is primary interest (e.g., communication network surveillance). In the time-series link prediction problem, the time series link occurrences are used to predict link occurrence at a particular time. In this paper, we propose a new time series link prediction based on learning automata. In the proposed algorithm for each link that must be predicted there is a learning automaton and each learning automaton tries to predict the existence or not existence of the corresponding link. To predict the link occurrence in time T, there is a chain consists of states 1 through T-1 and the learning automaton passes from state 1 through T-1 to learn the existence or not existence of the corresponding link. Using three co-authorship data sets, we have demonstrated that time-series models of link occurrences achieve better link prediction performance with commonly used static graph link prediction algorithms.

Keywords: Social Network, Link Prediction, Time Series, Learning Automata.

Introduction

Many data mining tasks involve (sometimes implicitly) prediction of linkages among data objects. Examples of explicit link prediction problems include automatic web hyperlink creation, prediction of genetic or protein-protein interactions, and the record linkage problem. Other well-studied problems can be viewed as an implicit link prediction problem once the data are rendered with a network/graph representation. These data can be visualized as graphs, where a vertex corresponds to a person in some group and an edge represents some form of association between the corresponding persons (1) (2). The associations are usually driven by mutual interests that are intrinsic to a group. All link prediction methods address the following question: "Given a pair of nodes u and v in the current social network, how likely is it that u will interact with v in the future?". Link prediction is applicable to a wide variety of application areas. For example, in the area of Internet and web science, it can be used in tasks like automatic web hyper-link creation (3) and web site hyper-link prediction (4). In e-commerce, one of the most prominent usages of link prediction is to build recommendation systems (5) (6). It also has various applications in other scientific disciplines. For instance, in bibliography and library science, it can be used for deduplication (7) and record linkage (8); in Bioinformatics, it has been used in protein-protein interaction (PPI) prediction (9). In security related applications, it can be used to identify hidden groups of terrorists and criminals (2).

Prior work on link prediction has been primarily formulated based on a static network setup, where a partial network structure is known and the objective is to predict the hidden links. In such a static network, link occurrence is typically modeled as a one-time event and the primary interest is on the existence of the link. For example, one may be interested to know whether a customer will purchase a product in the future or whether an author will ever collaborate with another author in the future. But in many application settings that involve dynamic networks, the link occurrence is preferably modeled as a sequence of binary states or occurrence frequencies, rather than a single binary state regarding its presence (10). In these applications much richer information could be extracted from the frequency time series of the link occurrences, such as the periodic patterns and temporal trends of the communication intensities. Generally, link prediction methods based on the static graph representation fall short when the repeated occurrences of the links are of central interest and temporal patterns are the primary feature of the application domain. One of the solutions to overcome this problem is link prediction using time series data. This problem is known as time series link prediction that is the problem of "Given link data between x and y for times 1 through T, how much is the likelihood of link occurrence between x and y at time T + 1?" (11). In this paper, we propose a new time series link prediction method based on the learning automaton, called LA-TSLP, to predict the link occurrence using time series data. In the proposed method each test link is modeled as a time series and predicted using a optimization tool called Learning automata.

Learning automaton (LA) is an adaptive decision making unit that tries to learn the optimal action from a set of allowable actions by interacting with a random environment (12). In each step, it selects an action from its action-set. The action selection in the LA is based on a probability distribution over the action set. The selected action is applied to the environment and then a reinforcement signal is produced by the environment. LA updates the probability distribution of its actions according to both reinforcement signal and a learning algorithm and again chooses an action. These steps are repeated until LA converges to some action.

In this paper we propose a new method that uses a set of LAs to predict time series link occurrence in the future. This new method utilizes a team of LAs to produce the link prediction using time series link occurrences data instead of static link occurrences data. It uses one LA for each test link. Each LA determines either the corresponding link should be appeared or not in time T by using previous occurrences in the network. In the proposed algorithm, to predict a link occurrence in the time T, the corresponding LA passes from state 1 through state T-1 and in each state, the action probability distribution of each LA is updated according to the reinforcement signal that is generated from corresponding state. The set of environments in each state t to generate the reinforcement signal is the set of the adjacent matrix of time t, in addition to some prediction results of time t+1 that is calculated based on different similarity metrics from the adjacent matrix of time t. So each learning automaton uses time series data of link occurrence in additional to some similarity metrics sequentially to generate prediction in the next time. Because of the nature of LA, it tries to learn the optimal behavior and so the optimal relation between each two node in the test set. These steps are repeated until the action of each LA converges to some value. The experimental results show that the LA-TSLP is superior to the some static link prediction methods such as Common Neighborhood (CN), Jacaard Index, Preferential attachment (PA) and Adamic-Adar Index (AA) in term of accuracy and performance.

The rest of the paper is organized as follows. Section 2 reviews the relevant literature on link prediction and time-series analysis. The learning automata are described in the section 3. Section 4 introduces the proposed temporal link prediction algorithm based on learning automata. Section 5 presents the experimental study on predicting co-authorship data sets. Section 6 summarizes the main conclusion and discusses the future directions of our research.

Background and Related Work

Problem Formulation

As it defined in (10), the time-series link prediction problem is formally introduced as follows. Let V be the list of vertices, V ={1, 2, …, N}. A graph series is a list of graphs $\{G_1, G_2, \ldots, G_T\}$ corresponding to a list of symmetric adjacency matrices $(M_1, M_2, \ldots, M_T)$. Each M_t is an $N \times N$ matrix with nonzero elements $M_{T(i,j)}$ corresponding to edges in $E_{T(i,j)}$. The value of $M_{T(i,j)}$ is the indicator of occurrence or not occurence of the undirected edge (i, j) during the time period t, which can also be viewed as an {0, 1} label in G_T . Then in the time series link prediction, we try to predict the occurrence or not occurrence of the links in time T+1 using previous times $M_1, M_2, \ldots, M_T$.

Related Works

This section reviews the recent time series link prediction methods:

Reference (13) has proposed a method for time-aware link prediction. They developed a novel approach of graph based link prediction and integrated it with temporal information (evolutionary history). In particular, they have extended the local probabilistic model (14) to include time awareness. An empirical evaluation of techniques was performed over two collaboration networks DBLP and astro-ph (astrophysics) slice of ArXiv. They showed that time of interactions between entities is a dominant feature for ranking neighboring nodes, which decides the probability of future interaction with the central node.

The authors of (11) have extended the problem of temporal link prediction to the problem of periodic temporal link prediction. They focused that if the data has underlying periodic structure and given link data for T time steps, can they predict the links at time T + 1, T + 2, T + 3, etc.? In the proposed method they consider bipartite graphs that evolve over time and consider matrix and tensor-based methods for predicting future links. They have presented a weight-based method for collapsing multi-year data into a single matrix and showed how the well-known Katz method for link prediction can be extended to bipartite graphs. Then using a CANDECOMP/PARAFAC tensor decomposition of the data, they have illustrated the usefulness of exploiting the natural three-dimensional structure of temporal link data. They have been evaluated their method on the DBLP bibliometric in terms of link prediction performance and relative expenses.

Authors (15) have proposed new method referred as cross temporal link prediction, in which the links among nodes of different time frames are predicted. This method extends dimension reduction based link prediction method proposed in (16) by including time stamp in prediction. Facebook, twitter, email etc. is an asynchronous communication system because messages have a time stamp. Most of messages are generated by replying and forwarding of previous messages so that the cross sectional links among originating node and replier exist. For inferring unobserved asynchronous communications, authors proposed, cross-temporal locality preserving projection (CT-LPP) method in which object mapping in different time frames is performed by low-dimensional latent feature space. They experimented on DBLP bibliographic data for cross temporal entity resolution and showed accuracy improvement.

Reference (17) has proposed an approach for link prediction which considered the evolution of topological matrices as a time series problem and used forecasting model. Their approach initially consists of building time series for each pair of non-connected nodes by computing their similarity scores at different past times. Their method deploys a forecasting model on these time series and uses their forecasts as the final scores of the pairs. In order to accomplish this goal, each chosen similarity metric is applied to all no connected pairs of nodes in the network in different past times. Then, a time series is built for each pair, recording the values provided by the metric. Finally, a forecasting model is applied to the series in order to predict their next values.

In (10), a new hybrid link prediction approach with both the interalink time-series patterns and interlink dependencies, have been proposed. They first investigated a time-series model for link prediction, in which the occurrence of each link is modeled as an independent time series. Specifically, for each link, they built an autoregressive integrated moving average (ARIMA) model based on its past occurrence series. Such a model ignores any interlink correlation information, which is the main data pattern employed by the existing static graph link prediction methods. They show that this algorithm achieved performance comparable with as existing link prediction methods under the static graph representation with e-mail and co-authorship data. Then they investigated hybrid link prediction methods that combine the power of the time-series model in predicting repeated link occurrences and the ability of static graph link prediction methods to identify new link occurrences. Their experiments showed that such hybrid methods achieved significantly better performance than the time-series model and static graph methods alone.

In (18) a weighted approach for modeling the occurrence of time is used to generate a different similarity metric and finally in our previous research we have proposed a new link prediction method based on temporal similarity metrics and Continuous Action set Learning Automata (CALA) (19). The proposed method takes advantage of using different similarity metrics as well as different time periods. In the proposed algorithm, we have modeled the link prediction problem as a noisy optimization problem and use a team of CALAs to solve the noisy optimization problem. The obtained link prediction results show satisfactory of the proposed method for some social network data sets.

Learning Automata

A learning automaton is an adaptive decision-making unit that improves its performance by learning how to choose the optimal action from a finite set of allowed actions through repeated interactions with a random environment (12) (20) (21) (22) (23) (24). This tool has many applications in different areas (25) (26) (27) (28). The action is chosen at random based on a probability distribution kept over the action-set and at each instant. The given action is served as the input to the random environment and the environment responds the taken action in turn with a reinforcement signal. The action probability vector is updated based on the reinforcement signal 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 $E = \{\alpha, \beta, c\}$ where $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ represents the finite set of possible action for each learning automata, $\beta = \{\beta_1, \beta_2, \dots, \beta_m\}$, denotes the set of the values can be taken by the reinforcement signal, and $c = \{c_1, c_2, \dots, c_r\}$ denotes the set of the penalty probabilities, where the c_i is associated with the given action α_i. If the penalty probabilities are constant, the random environment is said to be a stationary random environment, and if they vary with time, the environment is called a non stationary environment. The environments depending on the nature of the reinforcement signal β 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 its random environment has been shown in Figure 1.

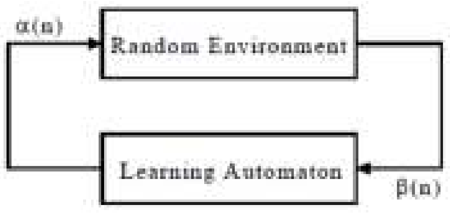


Figure 1 The relationship between the learning automaton and its random environment

Learning automata can be classified into two main families: fixed structure learning automata and variable structure learning automata. Variable structure learning automata are represented by a triple $< \beta, \alpha, T >$, where β is the set of inputs, α is the set of actions, and T is learning algorithm. The learning algorithm is a recurrence relation which is used to modify the action probability vector. Let α(k) and p(k) denote the action chosen at instant k and the action probability vector on which the chosen action is based, respectively. The recurrence equation shown by (1) and (2) is a linear learning algorithm by which the action probability vector p is updated. Let α_i(k) be the action chosen by the automaton at instant k. The updating rule when the taken action is rewarded by the environment (i.e., β(n) $= 0$) and when the taken action is penalized by the environment (i.e., β(n) = 1) is done based on the following equations (1), (2), respectively:

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

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

where r is the number of actions can be chosen by the automaton, a(k) and b(k) denote the reward and penalty parameters and determine the amount of increases and decreases of the action probabilities, respectively. If a(k) = b(k), the recurrence equations (1) and (2) are called linear reward-penalty (L_{R-P}) algorithm, if a(k) >> b(k) the given equations are called linear reward-ε penalty ($L_{R-\varepsilon P}$), and finally if b(k) = 0 they are called linear reward-Inaction (L_{R-I}). In the latter case, the action probability vectors remain unchanged when the taken action is penalized by the environment.

The Proposed Time Series Link Prediction Approach

This section proposes a new time series link prediction method based on learning automata called (LA-TSLP) that uses time series data of network structure for prediction task. This algorithm uses the network structure from time 1 through T sequentially to predict links in the time T+1. In the proposed algorithm there is a learning automaton for each test link in time T+1. In the proposed algorithm there are some stages that each stage corresponds to one time. In each stage t of the proposed algorithm each learning automaton tries to predict the link occurrence of the next time t+1. After finishing the prediction task in time t, the estimated prediction goes to the next stage. In the next stage each learning automaton tries to update and improve its estimation using a new environment t+1 to predict the link occurrence in time t+2 and so on. Also, in each iteration of the proposed algorithm there is two main phases: 1) training phase: an iterative phase to learn learning automata and 2) test phase: a phase to generate the output of the proposed method. The training phase in each stage has two steps: 1) Select the action of each learning automata and use it as the indicator of existence or not existence of the corresponding link, 2) update the action probability distribution of each learning automata based on a reinforcement signal the is produced from a time series environment. In the action selection phase of stage t in the proposed algorithm, each learning automata determines whether the target link is occurred in time t+1 or not. This action selection is done based on the internal action probability distribution of the learning automata. The probability of choosing action 1 for each learning automata in the start up of the algorithm is set to 0.5, $\mathrm{p}(1) = 0.5$. After choosing the action of LAs, a reinforcement signal is produced for each learning automata based on the effectiveness of selected action in the environment of time t. The set of environments in each state t to generate the reinforcement signal is the set of the adjacent matrix of time t, in addition to some predictions of time t+1 that is calculated based on different similarity metrics from the adjacent matrix of time t. So each learning automaton uses time series data of link occurrence in additional to some similarity metrics sequentially to generate prediction in the next time. Finally each learning automata updates its action probability distribution based on the received reinforcement signal and a learning algorithm. In the next iteration, each learning automaton again selects a new action based on the new action probability distribution and this procedure repeats until some stop criteria is satisfied. In the test phase for each test link, the converged action of the corresponding learning automaton is used as the prediction. The overall process of the LA-TSLP can be seen in Figure 2. The proposed algorithm is take advantage of using time series data of similarity metrics and occurrences data as well as using feedback of the previous time link occurrences. The pseudo code of the LA-TSLP can be seen in the Algorithm. 1. In the following sections, we describe two main phases of the proposed algorithm in detail.

Algorithm 1: Pseudo code of the proposed algorithm (LA-TSLP)

```
1:  Let $M_1, M_2, \ldots, M_T$ be the adjacent matrix from time 1 through T.
2:  Let n, t be the iteration counter and stage counter and initially set to **0** and let $Max_{NI}$ and T be the total number
    of iterations in each stage and total number of stages.
3:  Let E(t)=$\{\mathrm{M_t}, \mathrm{CN_t}, \mathrm{JA_t}, \mathrm{PA_t}, \mathrm{AA_t}\}$ be the set of environments for stage t.
4:  Let $a, b$ be the reward and punishment rate to update learning automaton in the learning algorithm.
5:  **Set** LAs be the set of learning automata indexes with 1 through J, one learning automaton for each link that must be
    predicted.
6:  **Set** the initial probability distribution of choosing action 1 for each learning automaton j to be $\mathrm{p_j}(1) = 0.5$.
7:  **while** $t<T$ do
8:      **while** $n< Max_{NI}$ do
9:          Select action j of each learning automaton $\mathrm{LA_j}$ based on the action probability distribution of the learning
            automaton, $\mathrm{p_j}$.
10:         Select one of matrixes in E(t) as C based on uniform probability a environment.
11:         Update the probability distribution function of each learning automata j, $\mathrm{p_j}$, based on the reinforcement
            signal that is generated from $C$ and learning algorithm $L_{R-P}$.
12:         **Set** $n=n+1$
```

```
13:     end while
14:         Set $t=t+1$
15: end while
16: For each test link j
17:         set $O_j = \alpha(LA_j)$ (converged action of $LA_j$)
```

Learning Phase

This sub section presents the training phase of the proposed algorithm. In this phase, there is a learning automaton for each test link that must be predicted. On the other hand there is a set of stages $\{S_1, S_2, \dots, S_T\}$, one stage for each time t. To do this, each learning automaton start in stage 1. In this stage the learning automata tries to learn the existence or not existence of the corresponding link in time 2 with the reinforcement signal that is produced from M_1. This prediction repeats until some termination criteria are satisfied. Then the learning automaton goes to stage 2. In stage 2, the learning automaton tries to update its learnt action using the new environment M_2 to predict the occurrence or not occurrence of the corresponding test link for time 3. The set of environment in stage S_t is the set of the adjacent matrix M_{t-1}, in addition to some prediction results for time t that is calculated based on different similarity metrics from the adjacent matrix M_{t-1}. So each learning automaton uses time series data of link occurrence as well as different similarity metrics sequentially to generate prediction in the next time and tries to learn the existence or not existence of the corresponding test link. These steps are repeated until the learning automata goes to time T and tries to learn the prediction in time T+1. To do this, the action set of each learning automaton LA_j is the set of integer numbers, {0, 1} and each learning automaton LA_j has a probability distribution p_j that determines the probability of choosing action 1. The probability of choosing action 1 for each learning automaton in the start up of the algorithm is set to 0.5, $p = 0.5$. So in the action selection step each learning automaton chooses action 1 and action 0 based on p_j and $1 - p_j$ as α_j, respectively. Action α_j is used as the prediction of corresponding link in time T+1. After action selection, these actions are evaluated based on a temporal environment and each learning automaton updates its probability distribution based on the reinforcement signal.

Now, to generate a reinforcement signal in stage t, we consider a set of environment: this set include adjacent matrix M_t, in addition to some predictions of time t+1 that is calculated based on different similarity metrics from the adjacent matrix M_t. In this paper we consider as Common Neighborhood (CN), Jacaard Index, Preferential attachment (PA) and Adamic-Adar Index (AA) as similarity methods. So the set of environment in the stage t is E(t)=$\{M_t, CN_t, JA_t, PA_t, AA_t\}$ that is the adjacent matrix, Common Neighborhood prediction result, Jacaard Index prediction result, Preferential attachment prediction result, and Adamic-Adar prediction result of time t, respectively. Then we select a matrix in E(t) as the environment of the learning automaton based on the uniform probability pt_i and call it C. Then for each learning automaton the corresponding value in adjacent matrix C is considered as the reinforcement signal β_j. Each learning automaton j finally updates its action probability distribution using the reinforcement signal β_j and L_{R-P} learning algorithm based on the following equation:

$$\begin{cases} p(n+1) = p(n) + a\big(1 - p(n)\big) & if\ \alpha_j = 1\ and\ \beta_j = 1 \\ p(n+1) = (1-a)\big(p(n)\big) & if\ \alpha_j = 0\ and\ \beta_j = 1 \\ p(n+1) = (1-b)\big(p(n)\big) & if\ \alpha_j = 1\ and\ \beta_j = 0 \\ p(n+1) = p(n) + b\big(1 - p(n)\big) & if\ \alpha_j = 0\ and\ \beta_j = 0 \end{cases} \tag{3}$$

Where $p(n)$, $p(n+1)$ is the probability distribution in iteration n and n+1, respectively. From equation () it follows that if action 1 is attempted in iteration n the probability distribution $p(n)$ is increased in iteration n for a favorable response and decreased for an unfavorable response. Also, if action 0 is attempted in iteration n the probability distribution $p(n)$ is decreased in iteration n for a favorable response and increased for an unfavorable response.

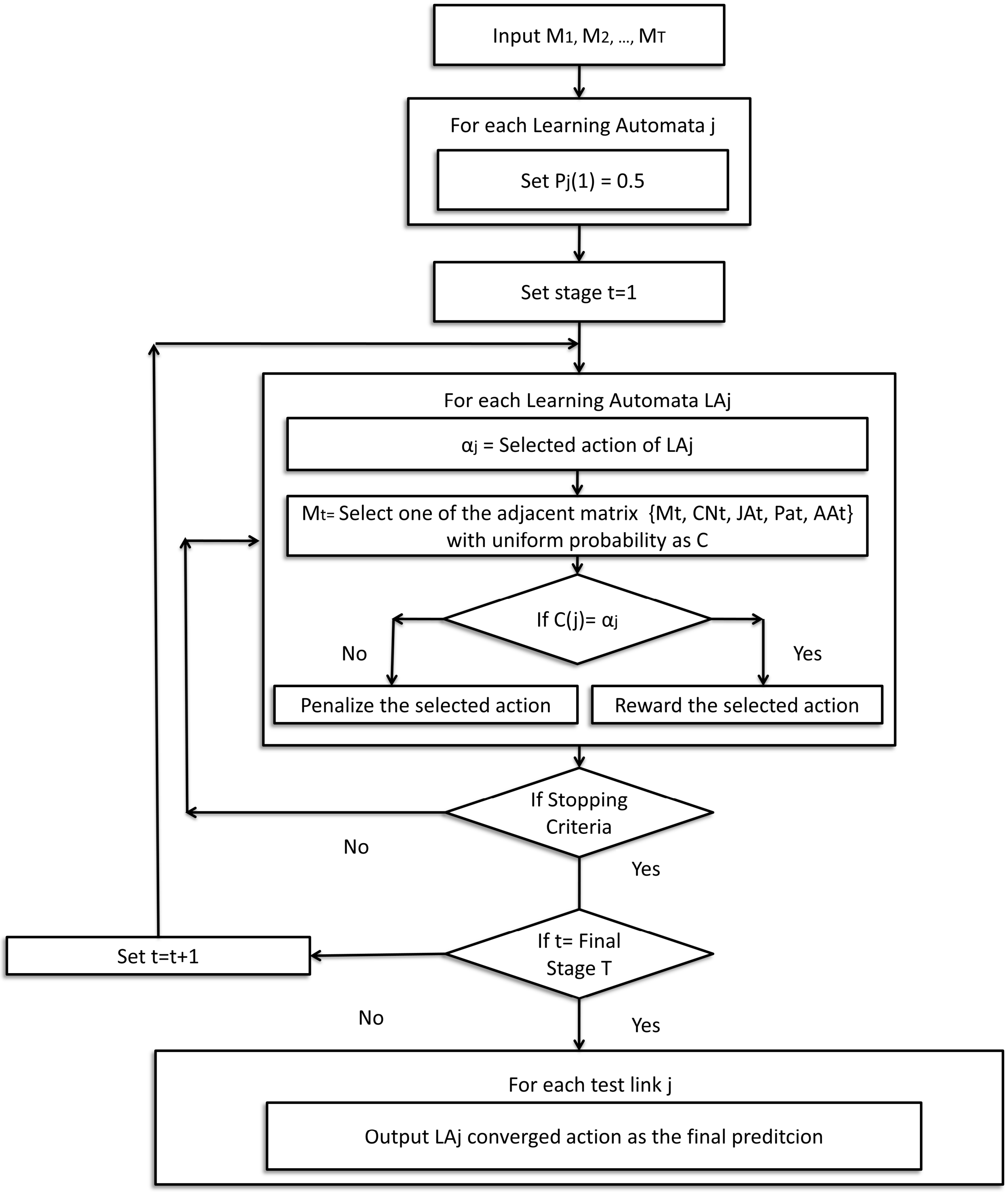
In the next iteration, each learning automaton selects a new action based on the new action probability distribution again and this procedure repeats until some stop criteria is satisfied. Then the learning automaton goes to the next step and the described procedures repeated for T stages. The prediction the last stage used as the final prediction for time T+1 based on the next following phase.

Prediction Phase

After running training phase, the test phase generates the final output prediction of the proposed algorithm. To do this, for each test link in time T+1, we set the final prediction to the converged LAj' action. In summary, the final prediction of a link j, O(j), is set based on the following rule:

$$O_j = \alpha(LA_j) \tag{4}$$

Where $\alpha(LA_j)$ and S_j is the converged $LA_j{}'$ action. Finally, the algorithm outputs matrix O as the output of proposed time-series link perdition method.



Input M1, M2, …, MT
For each Learning Automata j
Set Pj(1) = 0.5
Set stage t=1
For each Learning Automata LAj
αj = Selected action of LAj
Mt= Select one of the adjacent matrix {Mt, CNt, JAt, Pat, AAt} with uniform probability as C
If C(j)= αj
No
Yes
Penalize the selected action
Reward the selected action
If Stopping Criteria
No
Yes
If t= Final Stage T
Set t=t+1
No
Yes
For each test link j
Output LAj converged action as the final preditcion

Figure 2. The pseudo code of the proposed algorithm

Experiment Results

In order to evaluate the performance of the proposed algorithm, some computer experiments have been conducted and the performance of the proposed algorithm has been compared in term of performance and accuracy. In these experiments, we use the quality of solutions and the convergence rate of the proposed algorithm as the performance criteria. In the rest of this section, we first give the data set we used in our experiment and then give a set of three experiments. In the first experiment, the performance of the proposed algorithm compared with the performance of some static link prediction methods. In the second experiment, convergence rate of the LP-TSLA is computed and reported. Finally, in the third experiment, we study the sensitivity of the proposed algorithm on its parameters.

Data Set

In this section, we describe the social network data that we used in our experiments. For the experiments developed in this work, we adopted co-authorship networks, which are social networks where the nodes represent the authors, who are connected to each other if they collaborated in a paper. This kind of network is widely used to understand topology and dynamics of complex networks, since it corresponds to the largest publicly available digitalized social network. We adopted three co-authorship networks from three sections of Arxiv[1]. The first network is composed by authors that collaborated in theoretical high energy physics area[2] (hep-th). The second one is formed by authors who published papers in high energy physics area[3] (hep-ph) and the third on is sampled from collaboration authors in Astro Physics area[4] (Astro-ph).
(see Table 1 for information about the size of the networks). We extracted data from the year 1993 to 2003 for all these data sets. In these data sets if an author i co-authored a paper with author j, the graph contains an undirected edge from i to j. If the paper is co-authored by k authors this generates a completely connected (sub)graph on k nodes.

Table 1 Network Size in Terms of Nodes and Edges

Data Set	**Nodes (Authors)**	**Edges (Collaborations)**	**No of link must be predicted**
Hep-th	9,877	51,971	4,389
Hep-ph	12,008	237,010	15,523
Astro-ph	18,772	396,160	22,860

[1] http://www.arxiv.org
[2] http://arxiv.org/archive/hep-th
[3] http://arxiv.org/archive/hep-ph
[4] http://arxiv.org/archive/Astro-ph

Since co-authorship networks are highly sparse, we need to reduce the number of candidate pairs in order to make computation feasible. Aiming to decrease, we choose only the ones that at least have two collaborations during 1993 through 2002. Also in the next sub section we briefly describe the static methods that we use them in the comparison of the proposed method.

Static Link Prediction

The baseline methods we use them as the environment and in the comparison are described in detail by Liben-Nowell and Kleinberg (29). Let $\Gamma(x)$ denote neighbors of the node x, we define the following measures:

1. Common Neighborhood (30): In this measure, two nodes, x and y, are more likely to have a link if they have many common neighbors. This score is defined as
$$CN(x,y) = |\Gamma(x) \cap \Gamma(y)| \tag{5}$$
2. Jaccard Index (31): This index was proposed by Jaccard over a hundred year ago, and is defined as:
$$Jaccard(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|} \tag{6}$$
3. Preferential Attachment (PA) (32) (33): The preferential attachment (PA) algorithm is motivated by the preferential attachment phenomena (34) discovered in a variety of real world complex systems. Under this algorithm, the link occurrence score is set to be the product of the degrees of the involved nodes and is defined as follows:
$$PA(x,y) = |\Gamma(x)| \times |\Gamma(y)| \tag{7}$$
4. Adamic-Adar Index (AA) (35): This index refines the simple counting of common neighbors by assigning the less-connected neighbors more weight and is defined as:
$$AA(x,y) = \sum\nolimits_{z\in\Gamma(x) \cap\Gamma(y)} \frac{1}{\log(\Gamma(z))} \tag{8}$$

Link Prediction Comparison

In this experiment, we consider 10 years from 1993 to 2002, one stage for each year and we generate predictions for years 1996 through 2003. To generate prediction for a link in year y we use a learning automaton consist of stages {1993, 1994, …, y-1} and therefore, the snapshot of year y is used to generate prediction. Also, for all conducted experiments, the parameters of the LA-TSLP are chosen empirically based on the sensitivity analysis given in the last experiment. Also the maximum iteration of the proposed algorithm for each stage is set to 2000. Then the link prediction results of the proposed algorithm (LP-TSLA), for years 1996 through 2003 is compared to the static graph link prediction methods based on by AUC metric. All of the used algorithms are implemented in MATLAB R2009a in a PC, which has a single CPU of Intel(R) Core(TM)2 Duo 3.33 GHz and a 2 GB memory. The obtained result and its comparison of the proposed algorithm for Hep-th, Hep-ph and Astro-ph data set are reported in **Table**

2, **Table 3**, and **Table 4**, respectively. Also, it should be mentioned that for the following tables given in this section, the best results are highlighted.

Table 2 AUC Measures of the Proposed Time Series Algorithm and the Static Methods on Hep-th Data Set

Method/Data Set	**CN**	Jaccard Index	PA	AA	LA-TSLP
1996	**0.7989**	0.6098	0.6052	0.7046	0.7534
1997	0.7955	0.6033	0.6197	0.7284	**0.8447**
1998	0.7904	0.6210	0.6158	0.7594	**0.8904**
1999	0.7876	0.6390	0.6269	0.7710	**0.9018**
2000	0.7811	0.6401	0.6317	0.7794	**0.9132**
2001	0.7790	0.6501	0.6388	0.7901	**0.9589**
2002	0.8041	0.6445	0.6321	0.7935	**0.9771**
2003	0.7945	0.6438	0.6400	0.7962	**0.9817**
Average	0.7913	0.6314	0.6262	0.7653	**0.9026**

Table 3 AUC Measures of the Proposed Time Series Algorithm and the Static Methods on Hep-ph Data Set

Method/Data Set	**CN**	Jaccard Index	PA	AA	LA-TSLP
1996	0.7989	0.6098	0.6052	0.7046	**0.8322**
1997	0.7955	0.6033	0.6197	0.7284	**0.8483**
1998	0.7904	0.6210	0.6158	0.7594	**0.8838**
1999	0.7876	0.6390	0.6269	0.7710	**0.9161**
2000	0.7811	0.6401	0.6317	0.7794	**0.9225**
2001	0.7790	0.6501	0.6388	0.7901	**0.9419**
2002	0.8041	0.6445	0.6321	0.7935	**0.9612**
2003	0.7945	0.6438	0.6400	0.7962	**0.9806**

Average	0.7913	0.6314	0.6262	0.7653	**0.9108**

Table 4 AUC Measures of the Proposed Time Series Algorithm and the Static Methods on Astro-ph Data Set

Method/Data Set	**CN**	Jaccard Index	PA	AA	LA-TSLP
1996	**0.7989**	0.6098	0.6052	0.7046	0.7064
1997	**0.7955**	0.6033	0.6197	0.7284	0.7734
1998	0.7904	0.6210	0.6158	0.7594	**0.7917**
1999	0.7876	0.6390	0.6269	0.7710	**0.8267**
2000	0.7811	0.6401	0.6317	0.7794	**0.8661**
2001	0.7790	0.6501	0.6388	0.7901	**0.8792**
2002	0.8041	0.6445	0.6321	0.7935	**0.8967**
2003	0.7945	0.6438	0.6400	0.7962	**0.9667**
Average	0.7913	0.6314	0.6262	0.7653	**0.8384**

The results reported here demonstrate that the proposed time series link prediction method is able to achieve an average better AUC measure than the CN, Jaccard Index, PA and AA. The results suggest that the prediction is better in term of accuracy that is due to that the proposed algorithm uses the chain of link occurrences information to generate prediction.

Running Diagram of the LA-TSLP

In this experiment to show the evolutionary process of the proposed algorithm for each prediction, the accuracy graph of the proposed algorithm for data set Hep-th, Hep-ph, and Astro-ph are presented in Figure 3, Figure 4, and Figure 5. As it shown these figures, in the start of each state the proposed algorithm operates more stochastic and worse that is due to the operating in a new environment, but after a little iterations it learns the new pattern that is the aggregation of the previous learnt patterns in addition to the new environment pattern. Also, Table 5 shows the percentage of the learning automaton that has converged in each stage.

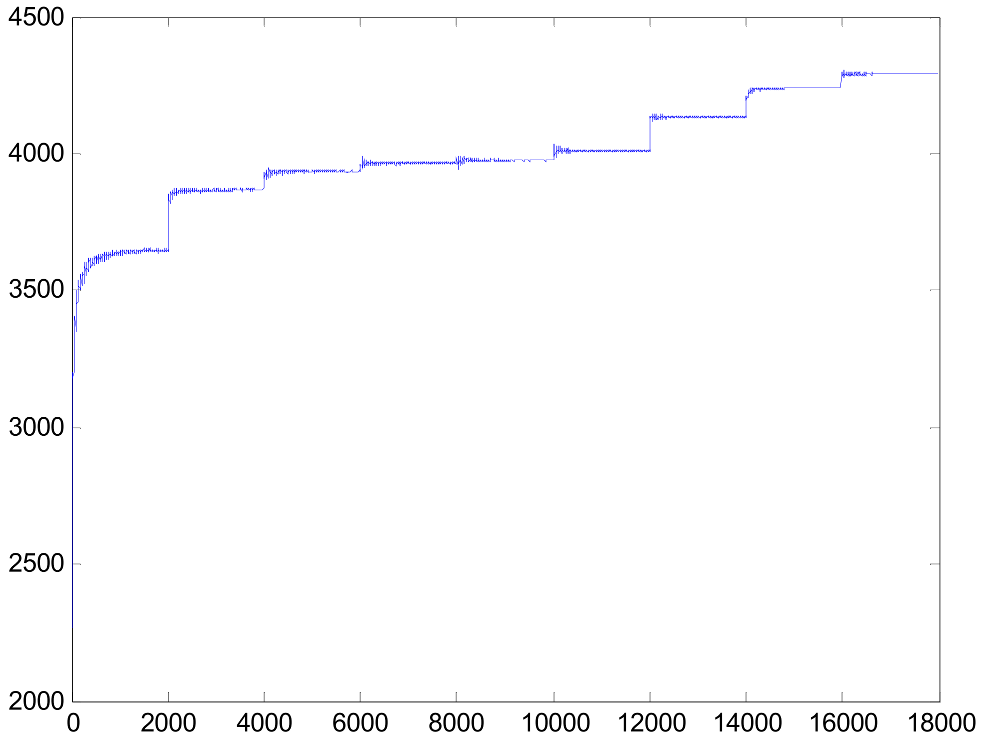


Figure 3 Running Diagram of the LA-TSLP for Hep-th Dataset

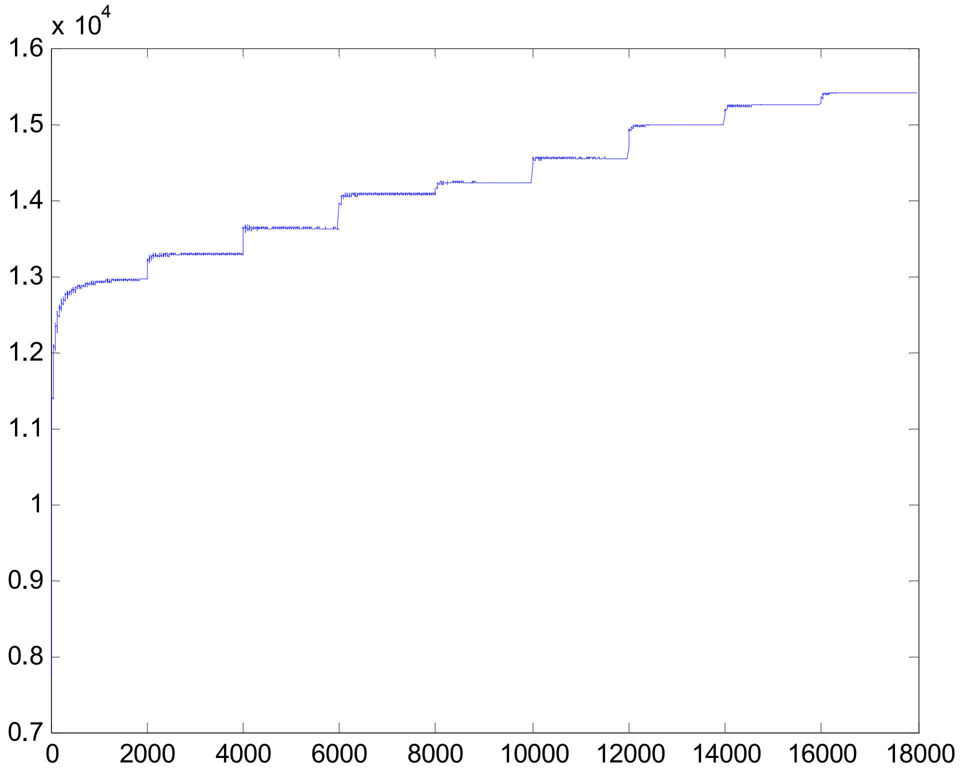


Figure 4 Running Diagram of the LA-TSLP for Hep-ph Dataset

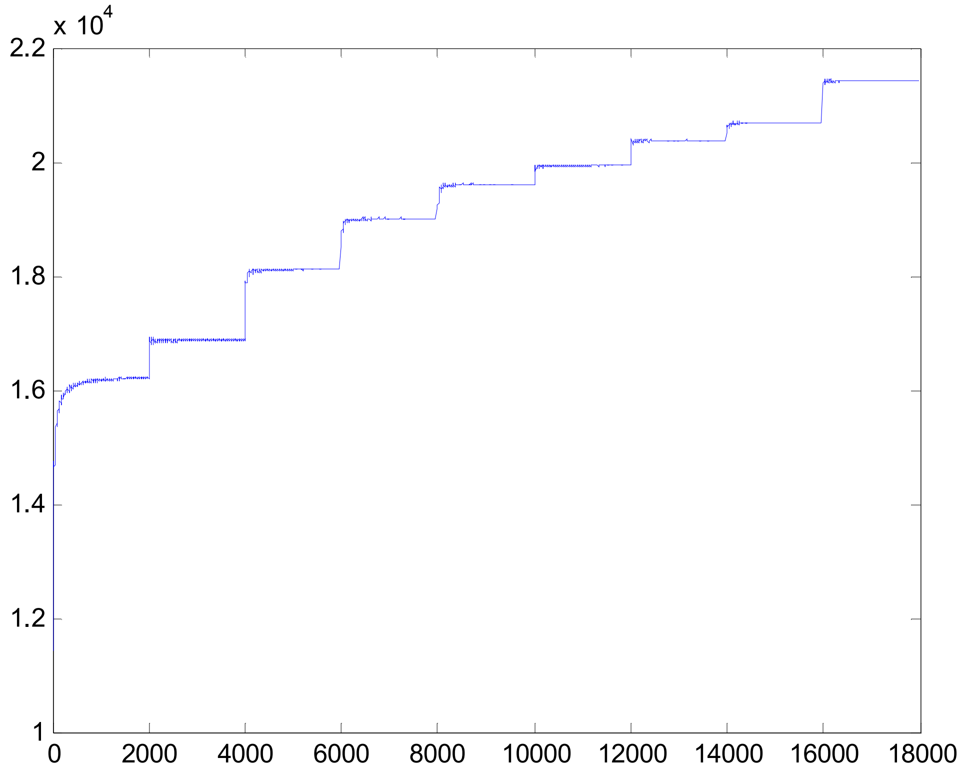


Figure 5 Running Diagram of the LA-TSLP for Astro-ph Dataset

Table 5 The Percentage of Converged Learning Automata through Learning Process

Year/Data Set	Hep-th	Hep-ph	Astro-ph
1996	0.9737	0.9847	0.9823
1997	0.9742	0.9869	0.9834
1998	0.9789	0.9882	0.9854
1999	0.9824	0.9899	0.9868
2000	0.9855	0.9915	0.9887
2001	0.9876	0.9935	0.9892
2002	0.9912	0.9950	0.9902
2003	0.9934	0.9978	0.9917
Average	0.9833	0.9909	0.9872

Sensitive analysis

In this section, we study the effect of parameter *a, b* in the learning algorithm: This experiment compares the impact of parameter *a, b* in the learning rate of the proposed algorithm. To do this, parameter *a*, b are varied from the set {0.001, 0.01, 0.02, 0.03, 0.04, 0.05} and the AUC measure of the proposed algorithm is calculated for each three data sets Hep-th, Hep-ph, and Astro-ph, and the results are presented in **Table 6**, **Table 7**, **Table 8**, respectively. These tables present that the best value of parameter *a*, b for all test data sets are about 0.01. This experiment shows that the lower learning rate causes later convergence and precisely accuracy in the performance of the LA-TSLP.

Table 6 AUC Measures of the Temporal Proposed Algorithm with Different Reward Rates for Hep-th Dataset

Year/Parameter	LA-TSLP (a=0.001) (b=0.001)	LA-TSLP (a=0.010) (b=0.010)	LA-TSLP (a=0.020) (b=0.020)	LA-TSLP (a=0.030) (b=0.030)	LA-TSLP (a=0.040) (b=0.040)	LA-TSLP (a=0.050) (b=0.050)
1996	**0.7534**	0.7482	0.6732	0.6723	0.6703	0.6705
1997	**0.8447**	0.8327	0.7634	0.7587	0.7523	0.7511
1998	**0.8904**	0.8865	0.8023	0.7982	0.7956	0.7912
1999	**0.9018**	0.8934	0.8212	0.8343	0.8325	0.8317
2000	**0.9132**	0.9067	0.8382	0.8362	0.8347	0.8302
2001	**0.9589**	0.9482	0.9021	0.9001	0.8973	0.8930
2002	**0.9771**	0.9693	0.9264	0.9232	0.9201	0.9184
2003	**0.9817**	0.9729	0.9313	0.9302	0.9294	0.9288

Table 7 AUC Measures of the Temporal Proposed Algorithm with Different Reward Rates for Hep-ph Dataset

Year/Parameter	LA-TSLP (a=0.001) (b=0.001)	LA-TSLP (a=0.010) (b=0.010)	LA-TSLP (a=0.020) (b=0.020)	LA-TSLP (a=0.030) (b=0.030)	LA-TSLP (a=0.040) (b=0.040)	LA-TSLP (a=0.050) (b=0.050)
1996	**0.8322**	0.8265	0.8007	0.8000	0.7997	0.7992
1997	**0.8483**	0.8294	0.8023	0.8017	0.8009	0.8001
1998	**0.8838**	0.8781	0.8539	0.8521	0.8502	0.8456
1999	**0.9161**	0.9077	0.8934	0.8912	0.8901	0.8890
2000	**0.9225**	0.9137	0.9014	0.9000	0.8991	0.8988
2001	**0.9419**	0.9356	0.9288	0.9256	0.9213	0.9201
2002	**0.9612**	0.9500	0.9478	0.9454	0.9433	0.9423
2003	**0.9806**	0.9612	0.9602	0.9589	0.9580	0.9569

Table 8 AUC Measures of the Temporal Proposed Algorithm with Different Reward Rates for Astro-ph Dataset

Year/Parameter	LA-TSLP (a=0.001) (b=0.001)	LA-TSLP (a=0.010) (b=0.010)	LA-TSLP (a=0.020) (b=0.020)	LA-TSLP (a=0.030) (b=0.030)	LA-TSLP (a=0.040) (b=0.040)	LA-TSLP (a=0.050) (b=0.050)
1996	**0.7064**	0.6843	0.6532	0.6503	0.6489	0.6402
1997	**0.7734**	0.7472	0.7231	0.7212	0.7187	0.7106
1998	**0.7917**	0.7687	0.7412	0.7399	0.7385	0.7352
1999	**0.8267**	0.8001	0.7832	0.7787	0.7723	0.7701
2000	**0.8661**	0.8412	0.8399	0.8387	0.8375	0.8360
2001	**0.8792**	0.8577	0.8423	0.8411	0.8403	0.8389
2002	**0.8967**	0.8723	0.8669	0.8608	0.8598	0.8581
2003	**0.9667**	0.9532	0.9459	0.9423	0.9412	0.9409

Conclusion

This paper presents a new time series link prediction method which uses learning automata to predict the occurrence or not occurrence of each link in time T+1 by using the network structure from time 1 to T. In the proposed method, for each link that must be predicted, there is one learning automaton and each learning automaton tries to learn the existence or not existence of the corresponding link in time T+1. The proposed algorithm has some stages and each stage corresponds to one time period. In each stage t of the proposed algorithm each learning automaton tries to predict the link occurrence of the next time t+1 by using similarity metrics and available link occurrence data. After finishing the prediction task in time t, the estimated prediction goes to the next stage. In the next stage each learning automaton tries to update and improve its estimation using a new environment t+1 to predict the link occurrence in time t+2 and so on. The experimental results reported here show that the proposed algorithm is superior to other static algorithms which consider only one snapshot of the network. The better result can be due to the learning capability of learning automata that shows that the occurrence of the link in the network evolves through time and a small modification of the previously occurrences as well as using different similarity metrics can be used for future predictions.

References

1. Lü, Linyuan, and Tao Zhou. "Link prediction in complex networks: A survey." Physica A: Statistical Mechanics and its Applications 390.6 (2011): 1150-1170.

2. Al Hasan, Mohammad, and Mohammed J. Zaki. "A survey of link prediction in social networks." Social network data analytics. Springer US, 2011. 243-275.

3. Adafre, Sisay Fissaha, and Maarten de Rijke. "Discovering missing links in Wikipedia." Proceedings of the 3rd international workshop on Link discovery. ACM, 2005.

4. Zhu, Jianhan, Jun Hong, and John G. Hughes. "Using Markov models for web site link prediction." Proceedings of the thirteenth ACM conference on Hypertext and hypermedia. ACM, 2002.

5. Li, Xin, and Hsinchun Chen. "Recommendation as link prediction: a graph kernel-based machine learning approach." Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries. ACM, 2009.

6. Huang, Zan, Xin Li, and Hsinchun Chen. "Link prediction approach to collaborative filtering." Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries. ACM, 2005.

7. Malin, Bradley, Edoardo Airoldi, and Kathleen M. Carley. "A network analysis model for disambiguation of names in lists." Computational & Mathematical Organization Theory 11.2 (2005): 119-139.

8. Elmagarmid, Ahmed K., Panagiotis G. Ipeirotis, and Vassilios S. Verykios. "Duplicate record detection: A survey." IEEE Transactions on knowledge and data engineering 19.1 (2007): 1-16.

9. Freschi, Valerio. "A graph-based semi-supervised algorithm for protein function prediction from interaction maps." International Conference on Learning and Intelligent Optimization. Springer Berlin Heidelberg, 2009.

10. Huang, Zan, and Dennis KJ Lin. "The time-series link prediction problem with applications in communication surveillance." INFORMS Journal on Computing 21.2 (2009): 286-303.

11. Dunlavy, Daniel M., Tamara G. Kolda, and Evrim Acar. "Temporal link prediction using matrix and tensor factorizations." ACM Transactions on Knowledge Discovery from Data (TKDD) 5.2 (2011): 10.

12. Thathachar, Mandayam AL, and P. Shanti Sastry. "Varieties of learning automata: an overview." IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 32.6 (2002): 711-722.

13. Tylenda, Tomasz, Ralitsa Angelova, and Srikanta Bedathur. "Towards time-aware link prediction in evolving social networks." Proceedings of the 3rd workshop on social network mining and analysis. ACM, 2009.

14. Wang, Chao, Venu Satuluri, and Srinivasan Parthasarathy. "Local probabilistic models for link prediction." Seventh IEEE International Conference on Data Mining (ICDM 2007). IEEE, 2007.

15. Oyama, Satoshi, Kohei Hayashi, and Hisashi Kashima. "Cross-temporal link prediction." 2011 IEEE 11th International Conference on Data Mining. IEEE, 2011.

16. Vert, Jean-Philippe, and Yoshihiro Yamanishi. "Supervised graph inference." Advances in Neural Information Processing Systems. 2004.

17. P. Ricardo, “Time Series Based Link Prediction,” pp. 10–15, 2012.

18. S. Huang, Y. Tang, F. Tang, and J. Li, “Link prediction based on time-varied weight in co-authorship network,” in Computer Supported Cooperative Work in Design (CSCWD), Proceedings of the 2014 IEEE 18th International Conference on, 2014, pp. 706–709.

19. Moradabadi, B., & Meybodi, M. R. (2016). Link prediction based on temporal similarity metrics using continuous action set learning automata. Physica A: Statistical Mechanics and Its Applications, 460, 361–373.

20. Narendra, Kumpati S., and Mandayam AL Thathachar. Learning automata: an introduction. Courier Corporation, 2012.

21. Thathachar, M. A. L., and P. Shanti Sastry. "A hierarchical system of learning automata that can learn die globally optimal path." Information sciences 42.2 (1987): 143-166.

22. Thathachar, M. A. L., and Bhaskar R. Harita. "Learning automata with changing number of actions." IEEE Transactions on Systems, Man, and Cybernetics 17.6 (1987): 1095-1100.

23. Thathachar, Mandayam AL, and Vijay V. Phansalkar. "Convergence of teams and hierarchies of learning automata in connectionist systems." IEEE Transactions on Systems, Man, and Cybernetics 25.11 (1995): 1459-1469.

24. S. Lakshmivarahan and M. A. L. Thathachar, "Bounds on the Convergence Probabilities of Learning Automata," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-6, pp. 756-763, 1976.

25. H. Beigy, M.R. Meybodi, A new continuous action-set learning automata for function optimization. J. Franklin Inst. 343, 27 (2006).

26. Santharam, G., P. S. Sastry, and M. A. L. Thathachar. "Continuous action set learning automata for stochastic optimization." Journal of the Franklin Institute 331.5 (1994): 607-628.

27. Oommen, B. John, and T. Dale Roberts. "Continuous learning automata solutions to the capacity assignment problem." IEEE Transactions on Computers 49.6 (2000): 608-620.

28. Obaidat, Mohammad S., et al. "Learning automata-based bus arbitration for shared-medium ATM switches." IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 32.6 (2002): 815-820.

29. Liben-Nowell, David, and Jon Kleinberg. "The link-prediction problem for social networks." Journal of the American society for information science and technology 58.7 (2007): 1019-1031.

30. Chen, Jilin, et al. "Make new friends, but keep the old: recommending people on social networking sites." Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2009.

31. Chowdhury, Gobinda. Introduction to modern information retrieval. Facet publishing, 2010.

32. Barabási, Albert-László, and Réka Albert. "Emergence of scaling in random networks." science 286.5439 (1999): 509-512.

33. Xie, Yan-Bo, Tao Zhou, and Bing-Hong Wang. "Scale-free networks without growth." Physica A: Statistical Mechanics and its Applications 387.7 (2008): 1683-1688.

34. Newman, Mark EJ. "Clustering and preferential attachment in growing networks." Physical review E 64.2 (2001): 025102.

35. Adamic, Lada A., and Eytan Adar. "Friends and neighbors on the web." Social networks 25.3 (2003): 211-230.