Knowledge-Based Systems 000 (2017) 1–10

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Trust propagation algorithm based on learning automata for inferring local trust in online social networks

Mina Ghavipour, Mohammad Reza Meybodi*

Department of Computer Engineering and IT, Amirkabir University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 28 December 2016
Revised 22 June 2017
Accepted 26 June 2017
Available online xxx

Keywords:
Trust
Trust propagation
Trust aggregation
Learning automata
Social networks

ABSTRACT

Online social networks have provided an appropriate infrastructure for users to interact with one another and share information. Since trust is one of the most important factors in forming social interactions, it is necessary in these networks to evaluate trust from one user to another indirectly connected user, using propagating trust along reliable trust paths between the two users. The quality of trust inference based on trust propagation is affected by the length of trust paths and also different aggregation strategies for combining trust values derived from multiple paths. While evaluating trust value based on all paths provides more accurate trust inference results, it is very time consuming to be acceptable in large social networks. Therefore, discovering reliable trust paths is always challenging in these networks. Another important challenge is how to aggregate trust values of multiple paths. In this paper, we first propose a new aggregation strategy on the basis of the standard collaborative filtering. We then present a heuristic algorithm based on learning automata, called *DLATrust*, for discovering reliable paths between two users and inferring the value of trust using the proposed aggregation strategy. The experimental results conducted on the online social network dataset of Advogato demonstrate that *DLATrust* can efficiently identify reliable trust paths and predict trust with a high accuracy.

1. Introduction

Social networks have been known as a popular medium for disseminating information and communicating with other people. Using such networks, users can connect to old friends, find new friends, or locate people who have the same problems or interests they have. In addition, companies can use social networks to improve their sales and deliver their services to customers. The problem with online social networks is that users have incomplete knowledge about their environment and other people, and they do not know how trustworthy the information on the internet is. As a result, trust plays a central role in online social networks.

Trust is an inherently interdisciplinary concept which is emerged from sociology [1,2], psychology [3,4], economics [5,6], and computer science [7,8]. Each of these disciplines has considered trust from different perspectives. A natural definition of trust in the context of a social web has been presented by Golbeck as “trust in a person is a commitment to an action based on a belief that the future actions of that person will lead to a good outcome” [9].

A lot of social applications, such as FilmTrust [10] and Epinions, use a web of trust to allow users to express their trust in other people. For people who have had no direct interactions, trust inference can be done via propagating trust through the network whenever there is at least one trust path between two users. That is, in the context of trust propagation approach, there are three basic operations for inferring the level of trust: discovering trust paths from a source user to an indirectly connected target user, propagating trust along each found path, and aggregating trust values derived from multiple trust paths. Based on this, three important factors in each trust propagation algorithm are: length of paths, propagation method, and aggregation method.

1.1. The motivation

When measuring the trust value between two users based on path discovery, the length of a trust path can become an issue [11]. Researchers [11,12] have been shown that the evaluating the trustworthiness of a target user based on all trust paths improves the quality of trust inference comparing to only shortest paths. However, finding all available paths between two users has the exponential time complexity. Since online social networks are usually massive in size, measuring trust using all paths becomes impractical in these networks. Various strategies have been considered in the literature to overcome this problem, such as: restrict-

* Corresponding author.
E-mail addresses: mina_ghavipour@aut.ac.ir (M. Ghavipour), mmeybodi@aut.ac.ir (M.R. Meybodi).

http://dx.doi.org/10.1016/j.knosys.2017.06.034
0950-7051/© 2017 Elsevier B.V. All rights reserved.

ing the depth of search [13,14], setting a limit to the search width [11,12,15]. However, these strategies are still time-consuming, as well as they reduce coverage of the trust propagation since some target users may become unreachable because of the limits on the search process. Moreover, the trust inference accuracy in these strategies decreases, since it is possible that the most reliable paths are lost as a result of blindly removing trust paths. Our motivation in this paper is to develop a new trust propagation algorithm based on learning automata that without considering any search constraints can address the above problem. Using learning automata helps our proposed algorithm to intelligently remove trust paths which are probable to be unreliable and (or) end in users other than the target user. In this way, because of ignoring only wasteful paths, the prediction coverage and accuracy in our proposed algorithm are maintained and in addition, the time complexity of the algorithm reduces. Learning automata has been successfully used in the literature for solving optimization problems, such as finding the shortest paths [16,17] which is similar to finding reliable trust paths in the trust propagation problem under consideration in this paper.

Another important issue in inferring the level of trust is the aggregation method. Two commonly used functions for aggregating trust values are: maximum and weighted average [12,18]. Investigating the performance of these functions for the trust inference has been indicated that the accuracy of using the weighted average function is higher than that of using the maximum function [11,12]. However, there are some weaknesses when using the weighted average function for trust aggregation. One problem is that the weighted average is not robust with respect to certain differences of interpretation of the trust scale. For example, for a trust scale from 1 to 10 (with 1 being *not trusted* at all, and 10 being *fully trusted*), one user may mistakenly think 1 means fully trusted and 10 means not trusted. In this case, a direct trust value of 1 from this user to the target user must result in a prediction of the trust value 10 for the source user toward the target user. Another example can be when one user uses only trust values between 6 and 10 for his neighbors and the source user only trust values between 1 and 6. As a result, a direct trust value of 10 from the first user to the target user must lead to a prediction of 6 for the source user toward the target user. However, these results will not be achieved by using the weighted average as the aggregation function. The weighted average function is not also robust against some malicious behaviors. As an example of a malicious behavior, a vicious user gives only low trust values to all of his neighbors [12]. Since these users attempt to obtain high trust values, their opinions highly affect the estimated trust value for the source user.

In addition, in the weighted average function, how much an opinion about the target user affects the final prediction not only depends on the reliability of the opinion owner, but also on the reliabilities of other opinion owners (since the weight of a direct trust provided by a neighbor of the target user is computed as the reliability of that neighbor divided by the sum of the reliabilities of all the target user's trusting neighbors). This makes a problem when all users trusting the target user have low reliabilities while expressing high trust values toward the target user. For example, using the weighted average aggregation function the estimated trust value from S to T in both Fig. 1(a) and (b) is 9. However, it is highly probable that in Fig. 1(a) the real trustworthiness of T for S is less than the estimated value in comparison with Fig. 1(b). This is while, in the domain of trust inference, clearly decreasing false positive is more important than decrease of false negative.

In this paper, we propose to utilize the standard collaborative filtering, with a little modification, for aggregating trust values of direct neighbors of the target user. The standard collaborative filtering is a well-known method for predicting the rating of a user on a target object based on his like-minded users [19,20].

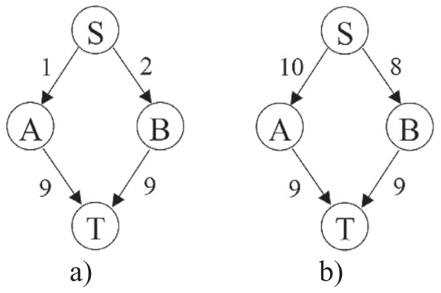

Fig. 1. Two examples of a trust network.

1.2. Our main contributions

The main contributions in this paper can be summarized as follows:

1. In order to overcome the weaknesses of the weighted average aggregation function, we propose a new strategy for aggregating trust values derived from multiple trust paths. This strategy is based on the standard collaborative filtering used for predicting a user's rating on a target object.
2. We also present a heuristic algorithm using distributed learning automata, called *DLATrust*, for discovering reliable trust paths and inferring the value of trust based on the developed aggregation strategy. The proposed algorithm assigns a learning automaton to each node of a trust network. The automaton in each node learns the next reliable nodes along trust paths toward the target node.
3. We conduct several experiments on the well-known trust network dataset: Advogato. The experimental results verify the effectiveness of our proposed aggregation method, and also indicate the efficiency and performance of the algorithm *DLATrust* for inferring the trust value between two indirectly connected users.

The rest of the paper is organized as follows. The next section discusses related work in the literature. In Section 3, a background of learning automata and distributed learning automata is briefly presented. Section 4 defines the problem that we address. Section 5 describes our proposed trust propagation algorithm for inferring the trust value. The experimental evaluation results are reported in Section 6. Eventually, Section 7 summarizes our work.

2. Background and related work

Trust is an important social concept which highly impacts users' decisions [21]. People use this concept of trust to help decide the extent to which they interact with others [22]. Based on this, decision support systems, as a tool to support decision making process, also use trust information among users to more effectively help them make their decisions in social networks. In particular, most existing successful recommender systems consider trust relations and recommend items to a target user from her trusted users [23]. It has been shown that incorporating trust into recommender systems can improve the quality and coverage of recommendations [24–27]. As a result, trust inference has been the focus of the great deal of attentions in recent years. Various research works have been carried out in this area. Several approaches have been proposed in the literature for measuring the trustworthiness of an unknown target user. One approach is based on trust propagation which estimates the level of trust from a source user to a target user by finding chains of trust relationships between them.

In this section, we first discuss each of three basic operations of a trust propagation strategy, including discovering trust paths, propagating trust along each path, and aggregating trust values from multiple paths, separately. After that, we briefly review the literature on trust propagation.

2.1. Discovering trust paths

The main objective in the trust inference by discovering trust paths is to maximize confidence in the estimation. Golbeck [9] observed that the accuracy of the trust estimation decreases as the length of trust paths increases and suggested that the shortest and strongest trust paths best estimate the value of trust. Authors in [28] pointed out the possibility that trust can be weakened or diluted through the propagation process. This is based on the fact that the longer the series of trusting users, the weaker is the estimated trust value. In contrast, authors in [29] stated that the trust value inferred from a long series of people with high trust may be much more confident than the trust value inferred from a short series of people with low trust. Authors in [30] defined two concepts of trust availability and path reliability and obtained the optimal length of a trust path based on a tradeoff between these two concepts.

Although considering all the paths from a source user to a target user to find the strongest of the possible paths shows improvement in the accuracy of trust prediction [11,12], but it has the time complexity of exponential [31]. For this reason, most research works in the domain of trust propagation whether confine the search process to shortest trust paths [9,18,21] or set a limit to the length of search [12–14], based on small-world network theory. Some other works, such as [11,15], by considering a minimum threshold for trust values during the process of finding the reliable paths among all paths reduce the number of visited users and thus reduce the computational cost of their algorithms. Authors in [32] propose a landmark based method with a preprocessing phase to improve the efficiency of trust propagation. Their proposed method firstly selects a small number of landmark users as referees in trust propagation, and then pre-computes the trust between the landmark users and the other users in the social network. Finally, the trust value between two indirectly connected users is predicted via aggregating the referrals provided by the landmark users.

2.2. Propagating trust along each path

Propagating trust along a path of trusting users is done based on the trust transitivity property, where if A trusts B and B trusts C, then it can be inferred that A trusts C to some extent [33]. Two widely used propagation functions are: *Min* and *Multi* [12,18]. Using the function *Min*, the propagated trust value is calculated as the minimum trust value among all the trust values along a path. In contrast, the function *Multi* computes the product of trust values along the trust path as the propagated trust value. While trust decays with the increase of the number of transitivity hops along a trust path [34], the function *Min* considers no decay in terms of the path length, but instead using the function *Multi* the decay is too fast when the length of path is relatively large [35]. Authors in [12] indicted that the accuracy of the propagation function *Min* for the trust inference has little difference with that of the propagation function *Multi*.

2.3. Aggregating trust values from multiple paths

The intuition behind aggregating multiple evidences of trust is that people naturally combine information obtained from different sources, assigning more credit to more trusted information sources. An aggregation method actually simulates this human ability and combines the trust information comes from parallel chains of trusting users to obtain a single trust value. Two aggregation functions are commonly used in the literature are: *Max* and *WAvg* [12,18]. The former takes the trust statement of the most trustworthy direct neighbor of the target user, while the latter takes the weighted average value among trust statements of all direct neighbors as the final estimated trust value. It has been shown that the aggregation function *WAvg* provides more accurate trust inferences than the aggregation function *Max* [12].

2.4. Related work

In the studies of trust propagation, Golbeck [9] proposed TidalTrust for establishing the trust relation between a source user and the target one based on averaging trust values along the strongest trust paths among all the shortest ones. The author has investigated the performance of the proposed algorithm with respect to simulated and actual trust networks. The SUNNY algorithm [21] uses a probabilistic sampling technique to estimate the confidence and computes the trust based on only those information sources with the highest confidence estimates. Actually, SUNNY executes the trust inference procedure from TidalTrust on a more confident sub network. Work in [36] described a trust inference algorithm called MoleTrust which discovers all shortest paths from the source user to a given target user and aggregates all direct trust values by calculating the weighted average. To improve accuracy, MoleTrust ignores any trust information from users whose trustworthiness is less than 0.6.

Shakeri and Bafghi [37] introduced an interval-based trust model to provide an integrated representation of trust and confidence. They also proposed two operators, named as trust interval Multiplication and summation, where the former propagates trust and confidence along all paths from a source user to a given target user and the latter aggregates two or more trust opinions. Work in [35] proposed a multi-dimensional evidence-based trust management system called MeTrust. The algorithm MeTrust uses *t*-norm to evaluate the trust for each path and the weighted average to combine the trust among multiple paths, where the weight of each path is deduced from the uncertainty of trust along the path.

Kim and Song [11] investigated the impact of the length of available trust paths and different aggregation methods on the accuracy of trust propagation. Authors presented four strategies for predicting the value of trust based on reinforcement learning and evaluated the prediction accuracy of those strategies: weighted mean aggregation among the shortest paths, min–max aggregation among the shortest paths, weighted mean aggregation among all paths, and min–max aggregation among all paths. They observed that the best is the combination strategy min–weighted mean among all trust paths. Kim also presented an enriched trust propagation approach based on this strategy in [15]. In his work, by combining a homophily-based trust network with an expertise-based trust network, he tackled the sparsity problem of the trust network. In this paper, we use learning automata, as a reinforcement learning approach, not only for learning the reliability of the direct neighbors of a target user (as done in [11]), but also for discovering trust paths between the source user and the target user.

3. Learning automata

A stochastic learning automaton [38,39] may be considered as an abstract model for adaptive decision making in an unknown random environment. The learning process of an automaton can be described as follows. At each instant, the automaton chooses an action from a finite set of allowable actions based on the probability distribution defined over the action set, and applies it to

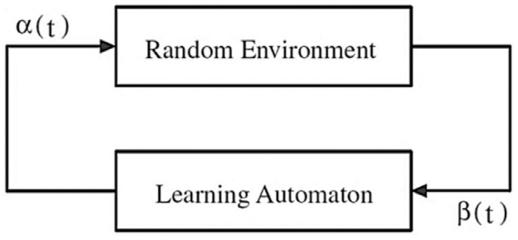

Fig. 2. The relationship between a learning automaton and its random environment.

the random environment with which the automaton interacts. The environment evaluates the chosen action and responses with a reinforcement signal with a certain probability. The automaton updates its action probability distribution depending on the signal received from the environment. By repeating these interactions, the automaton attempts to learn the optimal action which minimizes the probability of being penalized.

The random environment is described by a triple $\langle\alpha,\ \beta,\ c\rangle$, where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the input set of the environment, $\beta = \{\beta_1, \beta_2, \ldots, \beta_k\}$ represents the set of values taken by the reinforcement signal, and $c = \{c_1, c_2, \ldots, c_r\}$ is the set of penalty probabilities, where the element c_i corresponds to the input action α_i. If the penalty probabilities vary over time, the random environment is called a non-stationary environment and otherwise it is said to be a stationary environment. Random environments depending on the nature of the reinforcement signal can be categorized into *P*-model, *Q*-model and *S*-model. In *P*-model environments, the reinforcement signal can only take two binary values 0 and 1. *Q*-model environments allow a finite number of the values in the interval [0, 1] can be taken by the reinforcement signal. Finally, environments in which the reinforcement signal lies in the interval $[a, b]$ are referred to as *S*-model environments. The interaction of a learning automaton with its random environment is depicted in Fig. 2.

Stochastic learning automata can be classified into two main families: variable structure learning automata (VSLA) and fixed structure learning automata (FSLA). A variable structure learning automaton is represented by a quadruple $\langle\alpha,\ \beta,\ p,\ T\rangle$, where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ indicates the set of actions that automaton chooses from, $\beta = \{\beta_1, \beta_2, \ldots, \beta_k\}$ is automaton's set of inputs, $p = \{p_1, p_2, \ldots, p_r\}$ indicates the action probability vector, such that p_i is the probability of being chosen action α_i, and T is the learning algorithm which is used to update the action probability vector depending on the random environment's response, namely $p(t+1) = T[\alpha(t), \beta(t), p(t)]$, where inputs to the algorithm T are respectively the chosen action, the response of the environment and the action probability vector at instant t.

Let $\alpha_i(t)$ be the action chosen by the automaton and $p(t)$ denotes the action probability vector kept over the action set at instant t. $p(t)$ is updated as given in Eq. (1), if the chosen action $\alpha_i(t)$ is rewarded by the environment, and it is updated according to Eq. (2), if $\alpha_i(t)$ is penalized.

$$p_j(t+1) = \begin{cases} p_j(t) + a\big[1 - p_j(t)\big] & j = i \\ (1-a)p_j(t) & \forall j \neq i \end{cases} \tag{1}$$

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

where r denotes the number of actions that can be taken by the automaton, and a and b are the reward and penalty parameters which determine the amount of increases and decreases of the action probabilities, respectively. If $a = b$, the learning algorithm is a linear reward–penalty (L_{R-P}) algorithm, if $a \gg b$, it is a linear reward–ϵ penalty ($L_{R-\epsilon P}$) algorithm, and finally if $b = 0$, it is a linear reward–Inaction (L_{R-I}) algorithm in which the action probability vector remains unchanged when the taken action is penalized by the environment. The reward and penalty parameters a and b influence the speed of convergence as well as how closely the automaton approaches optimal behavior (the convergence accuracy) [39]. If a is too small, the learning process is too slow. In contrary, if a is too large, the increments in the action probabilities are too high and the automaton's accuracy in perceiving the optimal behavior becomes low. By choosing the parameter a to be sufficiently small, the probability of convergence to the optimal behavior may be made as close to 1 as desired [39]. In the $L_{R-\epsilon P}$ learning algorithm, the penalty parameter b is considered to be small in comparison with the reward parameter a ($b = \epsilon a$, where $0 < \epsilon \ll 1$). In this algorithm, the action probability distribution $p(t)$ of the automaton converges in distribution to a random variable p^* which can be made as close to the optimal vector as desired by choosing ϵ sufficiently small [40]. Stochastic learning automata has found many applications, such as graph sampling [41], fuzzy membership function optimization [24], vertex coloring [42].

3.1. Variable structure learning automata

A variable action set learning automaton is defined as an automaton in which the number of available actions at each instant varies over time. Such a learning automaton has a finite set of r actions, $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$. At each instant t, the action subset $\hat{\alpha}(t) \subseteq \alpha$ is available for the learning automaton to choose from. Selecting the elements of $\hat{\alpha}(t)$ is made randomly by an external agency. The procedure of choosing an action and updating the action probability vector in this automaton is done as follows.

Let $K(t) = \sum_{\alpha_i \in\ \hat{\alpha}(t)} p_i(t)$ presents the sum of the probabilities of the available actions in subset $\hat{\alpha}(t)$. Before choosing an action, the available actions probability vector is scaled as given in Eq. (3).

$$\hat{p}_i(t) = \frac{p_i(t)}{K(t)}\ \forall \alpha_i \in\ \hat{\alpha}(t) \tag{3}$$

Then, the automaton randomly chooses one of its available actions according to the scaled action probability vector $\hat{p}(t)$. Depending on the reinforcement signal received from the environment, the automaton updates the vector $\hat{p}(t)$. Finally, the available actions probability vector $\hat{p}(t)$ is rescaled according to Eq. (4). ε-optimality of this type of LA has been proved in [43].

$$p_i(t+1) = \hat{p}_i(t+1)\ K(t)\ \forall \alpha_i \in\ \hat{\alpha}(t) \tag{4}$$

3.2. Distributed learning automata

The full potential of learning automata is realized when they cooperate together to solve a particular problem. Distributed learning automata (DLA) [17] is a network of automata which are interconnected to work together. A DLA can be represented by a quadruple $\langle A, E, \mathcal{T}, A_0\rangle$, where A is the set of automata, $E \subset A \times A$ denotes the set of edges (the edge e_{ij} corresponds to the action α_{ij} of the automaton A_i), $\mathcal{T}$ is the set of learning algorithms based on which the learning automata update their action probability vectors, and A_0 denotes the root automaton from which the activation of the automata in DLA begins. Each time the root automaton is activated, it randomly chooses one of its outgoing edges (i.e. its actions) according to the corresponding action probability vector. As a result, the automaton on the other end of the chosen edge will be activated. The activated automaton also randomly chooses an action that results in the activation of another automaton. This process is repeated until a leaf automaton (i.e. an automaton that interacts with the environment) is activated. The chosen

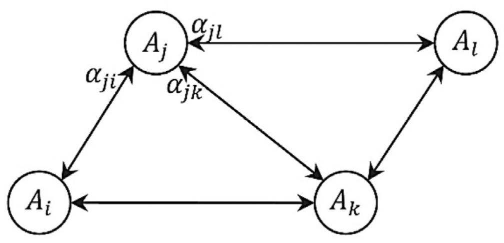

Fig. 3. Distributed learning automata.

Table 1
Notations description.

Symbol	Description
$G(V, E)$	a trust network
v_s/v_t	source/target node
v_i, v_j, v_k	a node in the trust network
$d_{out}(v_i)$	the out-degree of a node v_i
e_{ij}	a trust relationship from v_i to v_j
τ_{ij}	a direct trust value from v_i to v_j
$\bar{\tau}_k$	the average of trust values provided by v_k
τ'_{st}	the final estimated trust from v_s to v_t
ST_{sk}	the strength of a trust path from v_s to v_k
Nei_t	a set of direct neighbors of the target node
W_k	the reliability of a node v_k
Max_τ	the maximum value of trust

actions along the path induced by the activation process of the automata serve as the input to the environment. According to the response received from the environment, the activated learning automata update their action probability vectors. An example of DLA is shown in Fig. 3. DLA have repeatedly been used in literature to solve various problems [44–46]

4. Problem definition

Given a trust network as a directed weighted graph $G(V, E)$ with the set of nodes V and the edges' set E, a source node v_s, and an indirectly connected target node v_t. Let each edge $e_{ij} \in E$ be a trust relationship from the node v_i to the node v_j, with the weight τ_{ij} as the trust value. We assume that trust values between any two connected nodes are already known and represented by real values in interval $[0, 1]$, where 1 refers to full trust and 0 refers to no trust. Let Nei_t denotes the set of neighboring nodes of the target node v_t, who have expressed their trust values to v_t. We refer to these neighboring nodes as the direct neighbors of the target node and to their trust values as the direct trust values. The goal of a trust propagation problem is to infer the trustworthiness of v_t for v_s.

In this paper, we propose a trust propagation algorithm based on the reliability model which consider the two challenges of trust decay and path dependence [18]. Our proposed algorithm consists of two sub-tasks. It first determines the reliability of trust values provided by users directly trusting v_t. That is, the proposed algorithm find the most reliable trust path to each direct neighbor. The strength of that path computed based on a propagation function is considered as the reliability of the neighbor's direct trust. The final estimated trust value τ'_{st} is measured by aggregating the direct trust values weighted by their reliability values. The notations used in this paper are defined in Table 1.

5. DLATrust: DLA-based trust propagation

In this section, we present the details of our trust propagation algorithm based on distributed learning automata, called *DLATrust*. The input to the algorithm is a trust network $G(V, E)$, as well as a source node v_s and an unconnected target node v_t, and the output of the algorithm is the final estimated trust τ'_{st} from v_s to v_t. The trust propagation algorithm *DLATrust* given in Fig. 4 consists of three phases as follows.

5.1. Constructing DLA isomorphic to trust network

In this phase, a distributed learning automata isomorphic to the input trust network G is constructed. That is, each node v_i of G is equipped with a learning automaton A_i whose action set α_i includes the neighboring nodes who are directly trusted by v_i such that the size of α_i equals to $d_{out}(v_i)$. The activation process of the learning automata always starts from the automaton A_s corresponding to the source node v_s (as the root automaton) and in each time instant, only one automaton can be active. The activated automaton chooses one of its actions based on the action probability distribution, determining the next hop along the trust path towards the target node v_t. For each learning automaton, all actions have the same initial selection probability.

Since using opinions from multiple trust paths for the trust inference is better than using an opinion from a single trust path [12], in our proposed algorithm we attempt to discover reliable paths to a target node instead of only the most reliable path. For this purpose, we utilize the learning algorithm $L_{R-\epsilon P}$ for updating the action probability distribution of each automaton. As mentioned in Section 3, the learning algorithm $L_{R-\epsilon P}$ is an ergodic scheme, namely the action probability distribution $p(t)$ of the automaton converges in distribution to a random variable p^* which is independent of the initial probability distribution $p(0)$, instead of converging to an action [39]. In this way, each automaton learns the more trusted neighboring nodes, not only the most trusted one, along paths towards the target node.

5.2. Learning reliability of direct trust values

This phase attempts to discover the reliable paths from the source node v_s to the target node v_t and learn the reliability of the direct neighbors of v_t. The iteration t of the proposed algorithm *DLATrust* is described in the three following steps:

Step 1. Discovering a trust path
This step aims to find a trust path between the source node v_s and the target node v_t by a chain of automaton activations in DLA as follows. At first, the root automaton A_s (i.e. the automaton corresponding to the source node v_s) is activated to select the next hop along the trust path. The set of available actions for this automaton includes v_s's directly trusted neighbors whose corresponding automata have not been already activated along the current path. If there is no available action for the activated automaton A_s, the trust path cannot continue forward and thus this step is terminated. Otherwise, A_s chooses one of its available actions, say v_k, based on the scaled action probability distribution (for more details, see the Section 3.1). If the node v_k belongs to the set Nei_t (the set of direct neighbors trusting the target node v_t), a trust path towards the target node has been founded and this step can be successfully terminated. Otherwise, the automaton A_k corresponding to v_k is activated to have an action selection, and so on.

Step 2. Path evaluation
In this step, at first the strength of the trust path found in the previous step is computed based on the propagation function (for example, using the function Min the minimum trust value among all the trust values along the trust path is considered as the strength of the path). Then, the activated learning automata along the trust path receive a re-

```
Algorithm DLATrust : DLA-based trust propagation
01 Input  Trust network $G(V,E)$ , Source nodes $v_s$, Target nodes $v_t$
02 Output  Final estimated trust $\tau'_{st}$
03 Begin
     Phase1. Constructing DLA Isomorphic to Trust Network
04     Assign to each node $v_i$ of $G$ an automaton $A_i$ with $\boldsymbol{d_{out}(v_i)}$ actions
     Phase2. Learning Reliability of Direct Trust Values
05     Let $n_r$ be the number of rewards received by DLA which is initially set to 0
06     Set the reliability $W_i$ of each node $v_i$ to 0
07     Repeat
08       $A' \leftarrow$ automaton $A_s$ corresponding to the source node $v_s$
09       Let $path$ includes the traversed nodes along path from $v_s$ towards $v_t$ and is initially set to $v_s$
10       Repeat
11         Let the set of available actions for $A'$ includes all $v'$'s trusted neighbors except those in $path$
12         If the available action set is not empty then
13           $v_k \leftarrow$ automaton $A'$ chooses one of its available actions according to $\hat{p}(t)$
14           $path \leftarrow path \cup v_k$
15           $A' \leftarrow A_k$
16         End
17       Until ( There is no available action or $v_k \in \boldsymbol{Nei_t}$ )
18       $ST_{sk} \leftarrow$ compute the strength of $path$ based on a propagation function
19       If $v_k \in \boldsymbol{Nei_t}$ and $ST_{sk} \geq W_k$ then
20         Reward all the automata corresponding to the nodes in $path$
21         $W_k \leftarrow ST_{sk}$
22         $n_r \leftarrow n_r + 1$
23       Else
24         Penalize all the automata corresponding to the nodes in $path$
25         $n_r \leftarrow 0$
26       End
27     Until ( $n_r = \mathcal{R}$ )
     Phase3. Aggregating Direct Trust Values
28     Compute the final estimated trust $\tau'_{st}$ based on a aggregation function
29 End Algorithm
```

Fig. 4. The pseudo code of the *DLATrust* algorithm.

ward if the found path ends to a neighboring node v_k belonging to Nei_t and the strength of the path ST_{sk} is larger or equal to the maximum strength W_k already obtained for that node, and otherwise they are penalized. On receiving a reward, the maximum strength W_k for the ending node v_k will be updated to the strength of the found path.

Step 3. Stop condition

Two previous steps are repeated until DLA is rewarded for $\mathcal{R}$ sequential times, where $\mathcal{R}$ is a threshold for the sequential received rewards. In this situation, the maximum strength values obtained for the neighboring nodes of v_t are considered as the reliabilities of the direct trust values provided by those nodes and constitute the output of this phase.

5.3. Aggregating direct trust values

In this phase, the direct trust values are combined to obtain the final estimated trust, considering the reliability values computed in the previous phase. For this purpose, we propose to use the standard collaborative filtering, which is a well-known method for predicting the rating of a user v_s on a target object I_t based on his like-minded users [19,20], while making a little modification. Based on the standard collaborative filtering, the trust value from a source user v_s to a target user v_t is predicted as given below.

$$\tau'_{st} = \bar{\tau}_s + \frac{\sum_{v_k \in Nei_t} W_k \ (\tau_{kt} - \bar{\tau}_k)}{\sum_{v_k \in Nei_t} W_k} \tag{5}$$

Using the above equation for aggregating the direct trust values, two first weaknesses of the weighted average function, namely being un-robust against different interpretations of the trust scale and malicious behaviors, will be solved [47]. However, this equation still suffers from the problem of dependency of a direct trust's weight on the reliabilities of all the neighbors trusting the target user. In order to overcome this problem, we modify the above equation as follows:

$$\tau'_{st} = \bar{\tau}_s + \frac{\sum_{v_k \in Nei_t} W_k \ (\tau_{kt} - \bar{\tau}_k)}{|\, Nei_t |\ Max_\tau} \tag{6}$$

where $|Nei_t|$ indicates the number of the neighboring nodes in the set Nei_t. Based on this equation, the weight of a direct trust is computed as the proportion of the reliability of the direct trust to the maximum reliability (which is equal to the maximum value of trust Max_τ). In the experiments, we refer to the aggregation function given in Eq. (5) as *CFAvg* and to the function in Eq. (6) as *MCFAvg*.

6. Experimental evaluation

In this section, we conduct several expensive experiments to evaluate the efficiency of the aggregation functions *CFAvg* and *MCFAvg*, and also the performance of our proposed method *DLATrust*. The experiments are done on the real trust network dataset of Advogato, since they are among the most often-used datasets for evaluating trust inference performance [12,13,18]. Experimental results are averaged over 10 independent runs.

6.1. Experimental design

6.1.1. Evaluation method

The evaluation is done using a standard evaluation technique: leave one out [48]. At first, the trust relationship between the source node v_s and the target node v_t is masked. Then, the trust value from v_s to v_t is predicted based on a trust propagation algorithm using the remaining trust relationships. Finally, the predicted trust value is compared to the masked trust value as the actual trust.

6.1.2. Dataset

In order to evaluate the proposed trust propagation algorithm, we use the well-known trust network dataset Advogato (http://www.advogato.org). Advogato [49] is an online community and social networking website for developers of free software. One of its intended purposes is to serve as a research test-bed for testing trust metrics, with the idea that users can certify each other on four different trust levels: Observer, Apprentice, Journeyer, and Master. The dataset we used in this paper is a snapshot collected at 09:15-2014-Jul-07 which contains 56, 461 trust statements stated by 7419 users. In order to have real-valued trust statements, we assigned 0.2 to Observer, 0.6 to Apprentice, 0.8 to Journeyer, and 1 to Master.

To produce a smaller subgraph from the trust network dataset Advogato, we utilize the graph sampling algorithm *random walk* with the sampling fraction of 0.2. The sampled subgraph contains 1483 users and 10, 661 trust statements.

6.1.3. Evaluation metrics

In our experiments, we consider two metrics: *coverage* and *prediction accuracy*, which have been commonly used in [12–14,18]. Coverage measures the percentage of trust relationships that are predictable, that is, for which an algorithm can find at least one path between two users to infer the trust value. Prediction accuracy refers to the ability of predicting whether a user will be trusted or not. Different metrics used in this paper for evaluating the accuracy include:

Table 2
Strategies for comparison.

Strategy	Propagation	Aggregation	Paths
Min-Max	Min	Max	All shortest
Multi-Max	Multi	Max	All shortest
Min-WAvg	Min	Weighted average	All shortest
Multi-WAvg	Multi	Weighted average	All shortest

Mean absolute error (MAE). Let E' be a set of edges whose trust can be predicted. The MAE computes the average of the difference between the actual value of trust and the estimated trust value in all edges belonging to E'.

$$MAE = \frac{\sum_{e_{ij} \in E'} \left|\tau'_{ij} - \tau_{ij}\right|}{|E'|} \tag{7}$$

In most cases, the purpose of predicting the trust value is making decision to interact the target user or not. Based on this, we consider a threshold 0.5 for deciding to trust or not, such that if the estimated trust value is equal or more than 0.5, we will trust and otherwise not. We utilize the metrics *precision* and *recall* to compare the accuracy of algorithms in making decision.

Precision. Let T_A be the number of edges in which v_s directly trusts v_t, and T_B be the number of edges in which v_s trusts v_t by the trust value estimated through the algorithm. Precision measures the fraction of users who are predicted to be trusted and are actually trusted ones.

$$Precision = \frac{T_A \cap T_B}{T_B} \tag{8}$$

Recall. This metric measures the fraction of users are actually trusted and are successfully predicted.

$$Recall = \frac{T_A \cap T_B}{T_A} \tag{9}$$

Fscore. Since there is a trade-off between precision and recall, the Fscore metric is utilized to compute the accuracy using recall and precision jointly.

$$Fscore = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{10}$$

6.1.4. Methods for comparison

In order to investigate the efficiency of the proposed aggregation function and also our algorithm *DLATrust*, we implement all four strategies in Table 2 based on the reliability model. These strategies have been commonly used in the literature as the comparison standard [12,18]. We also compare our proposed algorithm with the *TidalTrust* [9] and *MoleTrust* [36] algorithms, which are well known in the domain of trust propagation.

6.2. Experimental results

6.2.1. Experiment I

In this experiment, we examine the effectiveness of the aggregation functions *CFAvg* and *MCFAvg*. For this purpose, we implement four trust propagation strategies: *Min-CFAvg, Multi-CFAvg, Min-MCFAvg*, and *Multi-MCFAvg*, based on the reliability model and compare the results of them with those of the strategies listed in Table 2. The comparison is done in terms of the accuracy metrics: MAE, Precision, Recall and FScore, and results on the dataset Advogato is presented in Table 3.

From this table, we can see that the different propagation functions provide the similar trust inference results. The aggregation functions *Max* and *WAvg* have the almost similar MAE, however Fscore of *WAvg* is much better than *Max*. The accuracy of using *CFAvg* is higher than that of using *WAvg*. Finally, the aggregation function *MCFAvg* performs the best on Advogato dataset.

Table 3
Comparison between different aggregation functions on Advogato dataset.

Strategy	MAE	Precision	Recall	FScore
Min-Max	0.1374	0.9202	0.9365	0.9283
Multi-Max	0.1377	0.9204	0.9348	0.9275
Min-WAvg	0.1326	0.9639	0.9535	0.9586
Multi-WAvg	0.1327	0.9208	0.9439	0.9322
Min-CFAvg	0.1208	0.9639	0.9535	0.9586
Multi-CFAvg	0.1209	0.9639	0.9532	0.9585
Min-MCFAvg	0.1177	0.9634	0.9649	0.9642
Multi-MCFAvg	0.1175	0.9633	0.9666	0.9650

Table 4
The impacts of learning parameters a and b on DLATrust' efficiency.

Setting		MAE	Precision	Recall	FScore
a	b				
0.09	0.0009	0.1177	0.9633	0.9702	0.9667
0.05	0.0005	0.1165	0.9634	0.9713	0.9673
0.01	0.0001	0.1152	0.8255	0.9724	0.9677
0.009	0.00009	0.1152	0.9632	0.9738	0.9684
0.005	0.00005	0.1149	0.9633	0.9729	0.9681
0.001	0.00001	0.1146	0.9634	0.9727	0.9680

6.2.2. Experiment II

This experiment aims to investigate the effects of the learning parameters a and b on the performance of our proposed algorithm *DLATrust*. We consider the different settings for these parameters and for each setting, we report the results of the accuracy metrics on Advogato in Table 4. The algorithm *DLATrust* is implemented by using the propagation function *Min* and the aggregation function *MCFAvg*, since *Min* provides the similar results as *Multi* and *MCFAvg* have the highest accuracy comparing to the other aggregation functions, considering the results of the previous experiment.

Based on the results of this table, decreasing the learning parameters always results in the decrease of MAE of the trust inference. However, for the metric FScore the decrease of the learning parameters at first increases and then decreases the value of the metric. Since for the setting $a = 0.009$ and $b = 0.00009$ the algorithm *DLATrust* reaches the maximum FScore over the Advogato dataset, for the experiments that follow we consider this setting for the learning parameters in our proposed algorithm.

6.2.3. Experiment III

In this experiment, we evaluate the efficiency of the proposed algorithm *DLATrust* in inferring trust value compared to other trust propagation algorithms in terms of the accuracy metrics. The algorithms for comparison are: *TidalTrust*, *MoleTrust*, as well as *Min-MCFAvg*. While *DLATrust* considers no limit for the trust path length, these three strategies are based on shortest paths from source to target. In order to investigate more accurately the efficiency of *DLATrust*, we also implement another version of the strategy *Min-MCFAvg*, called *Min-MCFAvg*^{AP}, in which all paths with the length less and equal to 7 between source and target are used for inferring the trust (Note that for the path length $\leq$ 7, the coverage of *Min-MCFAvg*^{AP} for the Advogato dataset is complete). The results of this experiment on the Advogato dataset are shown in Table 5. In addition to the accuracy metrics, we also report the execution time of the algorithms in this table, where the system on which the algorithms run has a hardware configuration of Intel® Core i5 2.53 GHz and 8 GB RAM.

According to the results, because of considering all trust paths the algorithm *DLATrust* obtains the higher estimation accuracy than the strategies using only shortest trust paths, though at the cost of time. In comparison with *Min-MCFAvg*^{AP}, *DLATrust* has a much smaller execution time, while FScore of both algorithms is almost similar. However, MAE of *DLATrust* is more than that of *Min-MCFAvg*^{AP}. The reason is that *DLATrust* pruns paths which are less likely to end in the target user with a high reliability and it is possible that some good paths are neglected in the search process.

Table 5
Comparison between different strategies on Advogato dataset.

Strategy	MAE	Precision	Recall	FScore	Time (s)
Min-MCFAvg	0.1177	0.9634	0.9649	0.9642	0.0374
Min-MCFAvg^{AP}	0.1129	0.9723	0.9662	0.9692	$8.1441E+04$
TidalTrust	0.1323	0.9210	0.9424	0.9316	0.0618
MoleTrust	0.1325	0.9207	0.9450	0.9327	0.1108
DLATrust	0.1152	0.9632	0.9738	0.9684	9.6957

6.2.4. Experiment IV

In this experiment, we study the effect of the number of opinions about target user on the trust inference accuracy. For different number of direct neighbors, we compare the average of the number of neighbors (or opinions) incorporated into the inference process, MAE and FScore of *DLATrust* with them of the other trust propagation algorithms including *TidalTrust, MoleTrust*, and *Min-MCFAvg*, respectively in Fig. 5(a), (b) and (c). From the results, we can see that the accuracy increases with the increase of the number of used opinions. It indicates that using more opinions from multiple paths provides better results when predicting trust. As shown in Fig. 5(a), because of considering all paths the algorithm *DLATrust* uses the more number of opinions and therefore, according to the results of Fig. 5(b) and (c), it has the higher accuracy comparing to other trust propagation algorithms.

6.2.5. Experiment V

This experiment aims to test the effect of the limitation of maximum path length on the performance of the proposed algorithm *DLATrust* comparing to other trust propagation algorithms. We varies the maximum length in the range of [2,6] and for each length, we report the accuracy in terms of MAE and FScore in Fig. 6(a) and (b), the execution time in Fig. 6(c), and the coverage in Fig. 6(d) for *DLATrust* and the other algorithms *TidalTrust, MoleTrust*, and *Min-MCFAvg*. Since for each maximum length the coverage is the same between the three algorithms *TidalTrust, MoleTrust*, and *Min-MCFAvg*, we report the value of coverage of these algorithms only once labeled as “Others” in Fig. 6(d).

From these figures, we can see that the coverage increases with increasing the maximum length, especially when it varies from 2 to 4. *DLATrust* has the same coverage as the other algorithms for all the maximum lengths. It indicates that this algorithm always successfully discovers trust paths satisfying the path length constraint between any pair of nodes. As shown in Fig. 6(c), the increase of the maximum length obviously increases the execution time. However, this increase is much faster for *DLATrust*. The reason is that *DLATrust* can find more paths as the maximum length gets larger.

In contrary, increasing the maximum length decreases the accuracy of the algorithms *TidalTrust, MoleTrust*, and *Min-MCFAvg*. As shown in Fig. 6(a) and (b), when the maximum length changes from 2 to 4, MAE of these algorithms increases and FScore decreases, while for the maximum length larger than 4, the values of these two metrics remain unchanged. We analyze the reason and find that for pairs of nodes whose shortest path length is equal to 3 or 4, the trust inference accuracy in these algorithms is low. Different results are observed for *DLATrust* when increasing the maximum length: FScore increases along with the maximum length, and MAE at first increases for the length of 3 and then decreases. The reason is that searching with a larger maximum length finds

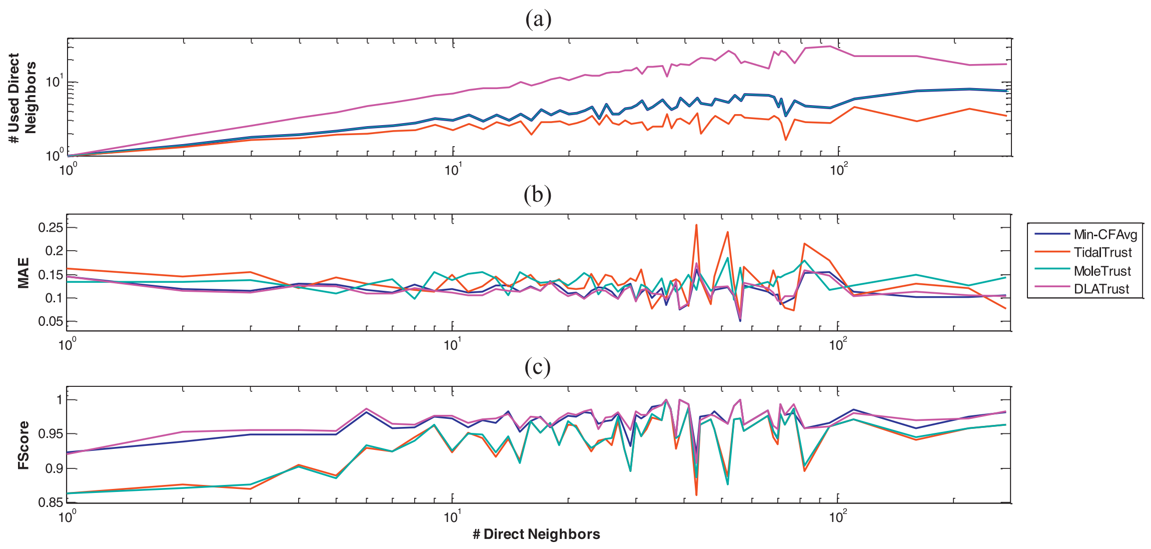

Fig. 5. The number of direct neighbors versus a) the average number of direct neighbors incorporated into the inference process, b) average MAE, c) average FScore. In general, the increase of the number of neighbors utilized increases the inference accuracy. Because of considering all paths, DLATrust uses more neighbors and thus obtains the higher accuracy.

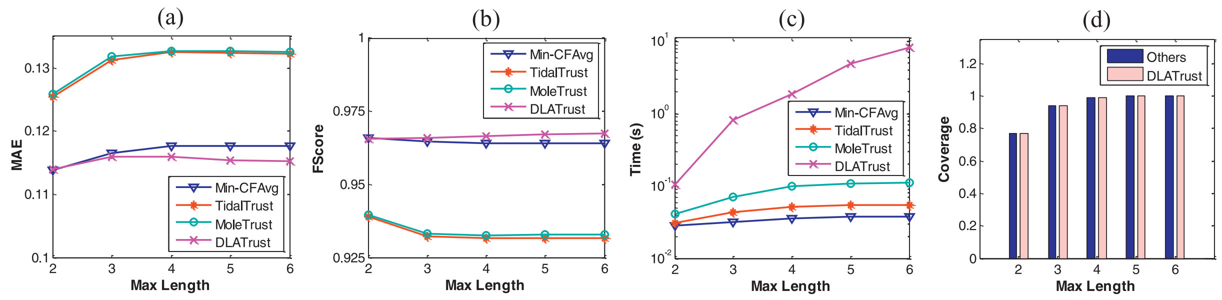

Fig. 6. The maximum length versus a) MAE, b) FScore, c) the execution time, d) coverage. In general, the increase of the maximum length increases coverage and the execution time. With increasing the maximum length, the accuracy for *DLATrust* increases and for the other strategies, conversely, decreases.

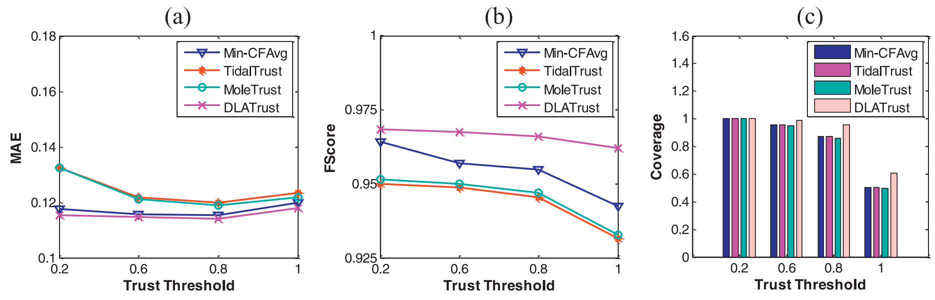

Fig. 7. The trust threshold versus a) MAE, b) FScore, c) coverage. In general, the increase of the trust threshold up to 0.8 decreases three measures. For the trust threshold of 1, FScore and coverage decrease more sharply, and MAE increases.

more trust paths, which means more opinions. It validates again that using more opinions results in a more accurate trust inference.

6.2.6. Experiment VI

In this experiment, we examine the effect of considering the trust threshold for the reliability of target's direct neighbors such that only the opinions of direct neighbors that are reliable at or above the trust threshold are incorporated in the trust inference process. For this purpose, the trust threshold is changed from 0.2 to 1 and for each threshold, we compare MAE, FScore and coverage of *DLATrust* with them of the other algorithms *TidalTrust, MoleTrust*, and *Min-MCFAvg*, respectively in Fig. 7(a), (b), and (c).

From the results, we can see that increasing the trust threshold decreases the coverage for all algorithms. However, the decrease is much slower in *DLATrust*, since it uses more opinions for inferring the trust value. During the increase of the trust threshold, FScore decreases sharply, and MAE also decreases for the trust threshold less or equal to 0.8 and increases for the threshold of 1. These results indicate that the trust threshold should not be set to a large value, since for a larger threshold more paths will be untrusted.

7. Conclusion

In this paper, we presented an efficient trust propagation algorithm using distributed learning automata to discover reliable trust paths and predict the trust value between two indirectly connected users. In this algorithm, an automaton is assigned to each node of the trust network to learn the next reliable nodes along trust paths towards the target node. In order to overcome the weaknesses of existing aggregation functions and improve the accuracy of trust inference, we also proposed a new aggregation function based on standard collaborative filtering for combining trust values derived from multiple paths. The experimental results on real trust network dataset show that our proposed algorithm can efficiently achieve high trust inference accuracy compared to existing trust propagation algorithms.

In future work, we will try to improve the proposed trust propagation algorithm, and also plan to design more extensive experiments to evaluate our proposed aggregation function on different trust networks.

References

[1] L.D. Molm, N. Takahashi, G. Peterson, Risk and trust in social exchange: an experimental test of a classical proposition, Am. J. Sociol. 105 (2000) 1396–1427. http://www.jstor.org/stable/3003770.
[2] G. Möllering, The nature of trust: from Georg Simmel to a theory of expectation, interpretation and suspension, Sociology 35 (2001) 403–420, doi:10.1177/S0038038501000190.
[3] J.B. Rotter, A new scale for the measurement of interpersonal trust, J. Pers. 35 (1967) 651–665, doi:10.1111/j.1467-6494.1967.tb01454.x.
[4] K.S. Cook, T. Yamagishi, C. Cheshire, R. Cooper, M. Matsuda, R. Mashima, Trust building via risk taking: a cross-societal experiment, Soc. Psychol. Q. 68 (2005) 121–142, doi:10.1177/019027250506800202.
[5] M. Granovetter, Economic action and social structure: the problem of embeddedness, Am. J. Sociol (1985) 481–510.
[6] F. Huang, Building social trust: a human-capital approach, J. Inst. Theory Econ. JITE 163 (2007) 552–573.
[7] D. Hughes, G. Coulson, J. Walkerdine, Free riding on Gnutella revisited: the bell tolls? IEEE Distrib. Syst. Online (2005) 1–18.
[8] M. Maheswaran, H.C. Tang, A. Ghunaim, Towards a gravity-based trust model for social networking systems, in: Twenty-Seventh Int. Conf. IEEE Distrib. Comput. Syst. Work. 2007. ICDCSW'07, 2007, p. 24.
[9] J.A.J.A. Golbeck, Computing and Applying Trust in Web-Based Social Networks, 2005, doi:10.1017/CBO9781107415324.004.
[10] J. Golbeck, J. Hendler, Filmtrust: movie recommendations using trust in web-based social networks, in: Proc. IEEE Consum. Commun. Netw. Conf., Citeseer, 2006, pp. 282–286.
[11] Y.A. Kim, H.S. Song, Strategies for predicting local trust based on trust propagation in social networks, Knowl. Based Syst 24 (2011) 1360–1371, doi:10.1016/j.knosys.2011.06.009.
[12] W. Jiang, G. Wang, J. Wu, Generating trusted graphs for trust evaluation in online social networks, Future Gener. Comput. Syst. 31 (2014) 48–58.
[13] P. Massa, P. Avesani, Trust metrics on controversial users: Balancing between tyranny of the majority, Int. J. Semantic Web Inf. Syst. 3 (2007) 39–64.
[14] S. Shekarpour, S.D. Katebi, Modeling and evaluation of trust with an extension in semantic web, Web Semantic Sci. Serv. Agents World Wide Web. 8 (2010) 26–36.
[15] Y.A. Kim, An enhanced trust propagation approach with expertise and homophily-based trust networks, Knowl.-Based Syst 82 (2015) 20–28, doi:10.1016/j.knosys.2015.02.023.
[16] S. Misra, B.J. Oommen, Dynamic algorithms for the shortest path routing problem: learning automata-based solutions, IEEE Trans. Syst. Man, Cybern. Part B. 35 (2005) 1179–1192.
[17] H. Beigy, M.R. Meybodi, Utilizing distributed learning automata to solve stochastic shortest path problems, Int. J. Uncertainty, Fuzziness Knowl.-Based Syst 14 (2006) 591–615.
[18] W. Jiang, J. Wu, F. Li, G. Wang, H. Zheng, Trust evaluation in online social networks using generalized network flow, IEEE Trans. Comput 65 (2016) 952–963.
[19] L. Lü, M. Medo, C.H. Yeung, Y.-C. Zhang, Z.-K. Zhang, T. Zhou, Recommender systems, Phys. Rep. 519 (2012) 1–49.
[20] X. Yang, Y. Guo, Y. Liu, H. Steck, A survey of collaborative filtering based social recommender systems, Comput. Commun. 41 (2014) 1–10.
[21] U. Kuter, J. Golbeck, Using probabilistic confidence models for trust inference in Web-based social networks, ACM Trans. Internet Technol 10 (2010) 8, doi:10.1145/1754393.1754397.
[22] E. Gray, J.-M. Seigneur, Y. Chen, C. Jensen, Trust propagation in small worlds, in: Int. Conf. Trust Manage., Springer, 2003, pp. 239–254.
[23] J. Tang, X. Hu, H. Liu, Social recommendation: a review, Soc. Netw. Anal. Min. 3 (2013) 1113–1133.
[24] M. Ghavipour, M.R.M.R. Meybodi, An adaptive fuzzy recommender system based on learning automata, Electron. Commer. Res. Appl. 20 (2016) 105–115, doi:10.1016/j.elerap.2016.10.002.
[25] M. Mao, J. Lu, G. Zhang, J. Zhang, Multirelational social recommendations via multigraph ranking, IEEE Trans. Cybern (2017).
[26] H. Wu, K. Yue, Y. Pei, B. Li, Y. Zhao, F. Dong, Collaborative topic regression with social trust ensemble for recommendation in social media systems, Knowl.-Based Syst 97 (2016) 111–122.
[27] W.-P. Lee, C.-Y. Ma, Enhancing collaborative recommendation performance by combining user preference and trust-distrust propagation in social networks, Knowl.-Based Syst 106 (2016) 125–134.
[28] A. Jøsang, E. Gray, M. Kinateder, Simplification and analysis of transitive trust networks, Web Intell. Agent Syst. Int. J. 4 (2006) 139–161.
[29] M. Lesani, N. Montazeri, Fuzzy trust aggregation and personalized trust inference in virtual social networks, Comput. Intell. 25 (2009) 51–83, doi:10.1111/j.1467-8640.2009.00334.x.
[30] J.-H. Cho, A. Swami, R. Chen, Modeling and analysis of trust management with trust chain optimization in mobile ad hoc networks, J. Netw. Comput. Appl. 35 (2012) 1001–1012.
[31] G.F. Liu, Y. Wang, M.A. Orgun, E.-P.P. Lim, Finding the optimal social trust path for the selection of trustworthy service providers in complex social networks, IEEE Trans. Serv. Comput. 6 (2013) 152–167, doi:10.1109/Tsc.2011.58.
[32] S. Lyu, J. Liu, M. Tang, Y. Xu, J. Chen, Efficiently predicting trustworthiness of mobile services based on trust propagation in social networks, Mobile Netw. Appl. 20 (2015) 840–852, doi:10.1007/s11036-015-0619-y.
[33] J. Golbeck, J. Hendler, Inferring trust relationships in web-based social network, ACM Trans. Internet Technol 6 (2006) 497–529.
[34] B. Christianson, W.S. Harbison, Why isn't trust transitive? in: Int. Work. Secur. Protoc., Springer, 1996, pp. 171–176.
[35] G. Wang, J. Wu, Multi-dimensional evidence-based trust management with multi-trusted paths, Future Gener. Comput. Syst. 27 (2011) 529–538, doi:10.1016/j.future.2010.04.015.
[36] P. Avesani, P. Massa, R. Tiella, A trust-enhanced recommender system application: Moleskiing, in: SAC, 2005, pp. 1589–1593.
[37] H. Shakeri, A.G. Bafghi, A confidence-aware interval-based trust model, ISC Int. J. Inf. Secur 4 (2013) 1–15.
[38] M.A.L. Thathachar, P.S. Sastry, Networks of Learning Automata: Techniques for Online Stochastic Optimization, Springer Science & Business Media, 2011.
[39] K.S. Narendra, M.A.L. Thathachar, Learning Automata: An Introduction (2012), doi:10.1109/TSMCB.2002.1049606.
[40] M.A.L. Thathachar, K.M. Ramachandran, Asymptotic behaviour of a learning algorithm, Int. J. Control 39 (1984) 827–838.
[41] A. Rezvanian, M.R. Meybodi, A new learning automata-based sampling algorithm for social networks, Int, J. Commun. Syst (2015) n/a-n/a, doi:10.1002/dac.3091.
[42] M.R. Mirsaleh, M.R. Meybodi, A new memetic algorithm based on cellular learning automata for solving the vertex coloring problem, Memetic Comput 8 (2016) 2112–2222, doi:10.1007/s12293-016-0183-4.
[43] M.A.L. Thathachar, B.R. Harita, Learning automata with changing number of actions, IEEE Trans. Syst. Man. Cybern. 17 (1987) 1095–1100.
[44] R. Forsati, M.R. Meybodi, Effective page recommendation algorithms based on distributed learning automata and weighted association rules, Expert Syst. Appl. 37 (2010) 1316–1330.
[45] J.A. Torkestani, M.R. Meybodi, Finding minimum weight connected dominating set in stochastic graph based on learning automata, Inf. Sci. (Ny). 200 (2012) 57–77.
[46] A. Rezvanian, M. Rahmati, M.R. Meybodi, Sampling from complex networks using distributed learning automata, Phys. A Stat. Mech. Appl. 396 (2014) 224–234, doi:10.1016/j.physa.2013.11.015.
[47] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, GroupLens: an open architecture for collaborative filtering of netnews, in: Proc. 1994 ACM Conf. Comput. Support. Coop. Work, ACM, 1994, pp. 175–186.
[48] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: IJCAI, 1995, pp. 1137–1145.
[49] R. Levien, A. Aiken, Attack-resistant trust metrics for public key certification., USENIX Secur., 1998.