# A Dynamic Algorithm for Stochastic Trust Propagation in Online Social Networks: Learning Automata Approach

Mina Ghavip our , Mohammad Reza Meybodi

PII: S0140-3664(17)31148-9
DOI: 10.1016/j.comcom.2018.04.004
Reference: COMCOM 5680

To appear in: *Computer Communications*

Received date: 1 November 2017
Revised date: 22 January 2018
Accepted date: 8 April 2018

# A Dynamic Algorithm for Stochastic Trust Propagation in Online Social Networks: Learning Automata Approach

Mina Ghavipour [a], Mohammad Reza Meybodi [b]

[a,b] *Department of Computer Engineering and IT, Amirkabir University of Technology, Tehran, Iran*

**Abstract:** The dynamic nature of trust has been universally accepted in the literature. As two users interact with each other, trust between them evolves based on their interaction's experience, in such a way that the level of trust increases if the experience is positive and otherwise, it decreases. Since social interactions in online networks, especially in activity and interaction networks, occur continuously in time, trust networks can be considered as stochastic graphs with continuous time-varying edge weights. This is while previous work on the trust propagation has assumed trust network as a static graph and developed deterministic algorithms for inferring trust in the graph. The problem becomes more challenging since trust propagation based algorithms are too time-consuming and therefore it is highly probable that trust weights change during their running time. In order to tackle this problem, this paper proposes a dynamic algorithm called DyTrust to infer trust between two indirectly connected users. The proposed algorithm utilizes distributed learning automata (DLA) to capture the dynamicity of trust during the trust propagation process and dynamically update the found reliable trust paths upon the trust variations. To the best of our knowledge, DyTrust is the first dynamic trust propagation algorithm presented so far. We conduct several experiments on the real trust network dataset, Kaitiaki, and evaluate the performance of the proposed algorithm DyTrust in comparison with the well-known trust propagation algorithms. The results demonstrate that by considering the dynamicity of trust, DyTrust can infer trust with a higher accuracy.

***Keywords:*** Trust, Trust propagation, Trust network, Social network, Stochastic network, Learning automata

## 1. INTRODUCTION

Online social networks provide an appropriate infrastructure for people interacting with each other without face-to-face interactions. Millions of people are joined these networks every day and interact with others who they did not know before. Due to some features of behaviors in social networks, such as anonymity, interactions are threatened seriously by vicious users. Trust is considered as one of the most effective methods to provide guidance for users identifying these vicious users or behaviors. Inferring trust among indirectly connected users plays a vital role in enforcing the security for social network sites and improving the quality of them.

Considerable number of studies has been done on trust in online social networks, and various methods have been proposed for inferring the trustworthiness of an unknown user. A popular family of these methods focuses on the trust inference via propagating trust through friendship networks whenever there exists at least one path between two users in these networks [1–10]. The problem with existing trust propagation-based methods is that they do not take into consideration the dynamic nature of trust. Trust is updated over time as a result of repeated interactions between users. Based on this, trust networks are stochastic graphs in which edge weights change rapidly and continuously in time. This is while, existing methods take an instant snapshot of trust network and then run a deterministic trust propagation algorithm on that snapshot. Since these methods are too time-consuming, it is highly probable that trust weights change during their running time and therefore the trust values estimated by them may become less relevant because of these trust variations.

In this paper, we address the dynamicity property of trust and propose a dynamic trust propagation algorithm based on distributed learning automata, called DyTrust, to infer the trustworthiness of an unknown target user while considering the changes of trust during its running time. Using learning automata, DyTrust first finds the most reliable trust path from the source user to each user directly trusting the given target. After that, the trust value of the source on the target user is estimated by propagating trust along the found reliable paths and then aggregating the opinions of the target's direct neighbors weighted by their reliability values. To the best of our knowledge, the proposed dynamic algorithm DyTrust is the first to consider the dynamicity of trust during the trust inference process and dynamically update the found reliable trust paths upon the trust variations. We conduct several extensive experiments on the real trust network dataset, Kaitiaki, and compare the performance of DyTrust with that of the well-known trust propagation algorithms, such as TidalTrust and MoleTrust. The results validate the effectiveness of our algorithm DyTrust: considering the changes of trust over time significantly improves the trust prediction accuracy.

The remainder of this paper is organized as follows. The section 2 surveys some related work in the literature of trust. In Section 3, learning automata and distributed learning automata are briefly reviewed. Sections 4 and 5 respectively present the details of our proposed dynamic trust propagation algorithm DyTrust and its convergence proof to the optimal solution. The experimental evaluation is described in Section 6. Eventually, Section 7 discusses and concludes our work.

## 2. RELATED WORK

### 2.1 Trust in Online Social Networks

Online social networks provide a convenient opportunity for people to interact with each other and share their information. Trust is considered as a vital factor in forming social interactions. Researchers have proposed various definitions for trust in different contexts. In the context of a social web, Golbeck [11] has defined trust in a person as "a commitment to an action based on a belief that the future actions of that person will lead to a good outcome". Therefore, reliability in previous interactions with a person give rise to positive expectations about that person's intentions [12]. As two users interact with each other, trust between them evolves based on their interaction's experience, in such a way that the level of trust increases if the experience is positive and otherwise, it decreases. Trust may also decay with time. Experiences of more recent interactions are given higher importance than those of old ones, since experiences of old interactions may become irrelevant with time [13]. This property refers to the dynamicity of trust.

Various techniques have been proposed for modelling this dynamicity, such as aging of old experiences [14–16], giving more weight to recent experiences [17–19], considering only the most recent experience [20,21], using temporal window [22–25] and periodically computing the value of trust [26,27].

### 2.2 Related Studies for Inferring Trust

Existing models for inferring trust are classified into two categories from the standpoint of trust propagation: non-propagating and propagating trust models. One of the famous models in the first category is PeerTrust [24], which was the first to introduce the concept of feedback credibility and its role for defending against dishonest feedbacks. PeerTrust included a coherent adaptive trust model to quantify and compare the trustworthiness of peers by using a transaction-based feedback system. Another model without trust propagation process was proposed by work in [28]. In this paper, Authors provided a trust model for virtual communities based on a reputation mechanism and direct experiences. Liu et al. [29] presented a supervised learning approach that automatically predicts trust among users of online communities using two factors: user factor and interaction factor, where the former refers to evidence derived from actions of individual users and the latter is interactions between pairs of users. Caverlee et al. [30] proposed the SocialTrust framework for supporting tamper-resilient trust establishment in online social networks. SocialTrust initially gives all users the same level of trust. Then, it dynamically revises trust ratings according to three components: the current quality of trust, the interaction history, and the adaptation to change. Authors in [31] proposed a decision support method for estimating trust in virtual teams. Their proposed trust estimation framework has two dimensions: Reputation and Collaboration, where the former represents the trustworthiness of members and the later represents the cooperation situations between members in a team. Works in [32–35] are also subcategorized in non-propagating classification, since they discussed trust inference models relying on past observed behaviors. As another example of trust model without propagation, game theory [36–38] has been utilized in recent years for inferring trust. Most of the work in this area has focused on designing some mechanism to make individuals cooperate with each other rather than defecting. Although there have been some attempts at using game theory for trust among agents [39,40], this approach is still not among the primary focuses of research in the scope of trust management.

Trust models based on propagation are more popular than non-propagating trust models. In the trust propagation approach, users propagate their trust value to others through trust network following their direct trust relationships. One of the most famous trust models with propagation has been proposed by Golbeck [11]. The author studied the concept of trust in web-based social networks and observed that the accuracy of trust inference decreases as the trust path length increases. Therefore, she suggested that the shortest and strongest paths are the best for the estimation of trust and presented a trust propagation model called TidalTrust for inferring the trust value of a source user related to a target one based on averaging trust values along the strongest shortest trust paths. Another famous model based on trust propagation is MoleTrust [41], which finds all shortest trust paths from a source to a target user and combines all direct trust values issued by users whose trustworthiness is more than 0.6. Authors in [42] investigated the impact of the path length and strength on the accuracy of trust inference and observed that the strength of a trust path can be more important than its length, such that stronger trust paths are favored to weaker but shorter ones. Although their fuzzy trust inference algorithm considering all strongest paths performs more precisely the inference process, it has the time complexity of exponential and therefore cannot be applicable to large scale social networks. Work in [43] described a trust inference algorithm called SUNNY, which estimates the confidence by using a probabilistic sampling technique and computes trust only based on the information sources with highest confidence estimates. Actually, this algorithm executes the trust inference procedure from the TidalTrust algorithm on a more confident sub network.

With a variation on the PageRank algorithm, Kamvar et al. [14] proposed a distributed trust propagation model called EigenTrust for estimating trust values in P2P networks. EigenTrust measures trust values through iterative multiplication and aggregation of trust values along transitive chains until the trust values for all agents converge to stable values. While the EigenTrust algorithm shows a satisfied performance on simple threat models, it could not offer good attack resilience when encountering more sophisticated threat ones. Authors in [44] analyzed the vulnerabilities of EigenTrust and proposed the trust model ServiceTrust which has a better performance on some sophisticated attack models by utilizing pairwise feedback similarity weighted trust propagation into the trust model. Kim and Song [7] studied the impact of two factors: the length of trust paths and aggregation functions, on the trust inference accuracy and proposed four strategies for estimating trust based on reinforcement learning. They found that the best combination is the strategy of weighted mean aggregation among all trust paths. Based on this strategy, Kim also proposed an enhanced trust propagation approach by combining a homophily-based trust network with an expertise-based one [1]. Using this approach, author tackled the sparsity problem of trust networks. Work in [45] presented a content-driven trust propagation framework that discovers credible claims and estimates trustworthiness of sources based on the quality of evidence. In fact, their proposed techniques for identifying and scoring relevant posts could be used to instantiate a model for computing the trustworthiness of medical claims and sources. Authors in [46] proposed a method called PIN-TRUST to measure the trustworthiness of each user for a target user by employing belief propagation (BP). PIN-TRUST exploits all kinds of interaction information, including explicit trust, implicit trust and explicit distrust, in the trust prediction process.

Another example of trust model with propagation is MeTrust [5], a multi-dimensional evidence-based trust management system. MeTrust uses the Frank t-norm and the weighted average to combine trust values respectively along a path and among multiple paths. In this model, the weight of a trust path is deduced from uncertainty of trust along the path. Work in [3] presented a trust propagation model which includes a landmark based method with a pre-processing. In this method, at the first a small number of landmark users are chosen as referees in trust propagation process, and trust between the landmark users and the others are precomputed. Then, the referrals provided by the landmark users are aggregated to estimate trust between two indirectly connected users. In our previous work on trust propagation [9], we proposed a heuristic trust propagation algorithm based on learning automata, called DLATrust, as well as a new aggregation strategy. We indicated that using our developed aggregation strategy the DLATrust algorithm can efficiently discover reliable trust paths without any search constraints and precisely infer the trust value between two indirectly connected users.

All previous works on the trust inference based on propagation have assumed trust network to be a static graph and developed deterministic algorithms for inferring trust in the graph. This is while, the dynamic nature of trust has been universally accepted in the literature. Trust values change rapidly and continuously in time as a result of repeated interactions between users. On the other hand, trust propagation based methods are too time-consuming and therefore it is highly probable that trust weights change during their running time. Under these conditions, these methods are not appropriate, since they consider the trust network at one point in time and thus the trust values estimated by them may become less relevant because of the trust variations. To the best of our knowledge, we are the first to address the dynamicity of trust during the trust inference process and propose a dynamic trust propagation algorithm, called DyTrust, which captures the temporal variations of trust weights and dynamically updates found reliable paths upon these variations. DyTrust is different from our previous work DLATrust [9] in two major aspects. First, DyTrust finds the most reliable trust path to each direct neighbor of a given target user. In this way, it uses the opinions of all the direct neighbors for inferring the final trust value. However, DLATrust attempts to discover reliable trust paths to the target user, which means not all opinions about the target are incorporated in the trust inference process. Therefore, DyTrust estimates more accurately the trustworthiness of the target user. Second, contrary to DyTrust, our previous work DLATrust does not take into consideration the changes of trust weights during the propagation process and thus its estimated trust values have not enough accuracy.

## 3. LEARNING AUTOMATA AND DISTRIBUTED LEARNING AUTOMATA

This section presents a brief review of learning automata and distributed learning automata.

## 3.1 Learning Automata

A stochastic learning automaton [47,48] is a reinforcement learning approach which attempts to learn the optimal action through trial-and-error interactions with an unknown random environment and improve its performance. At each discrete time $t$, the automaton selects an action $\alpha(t)$ from a finite set of given actions based on the probability distribution kept over the action set. The selected action serves as the input to the random environment with which the automaton interacts. As a consequence of its action $\alpha(t)$, the automaton receives an immediate response $\beta(t)$ (a reinforcement signal) from the environment, based on which it updates its action probability distribution. By these repeated interactions, the automaton converges to the optimal action which minimizes the received average penalty. The random environment can be described by a triple $\langle \alpha, \beta, c \rangle$, where

- $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ is the finite input set of the environment.
- $\beta = \{\beta_1, \beta_2, \dots, \beta_k\}$ represents the set of the values taken by the reinforcement signal.
- $c = \{c_1, c_2, \dots, c_r\}$ is the set of the penalty probabilities, where element $c_i$ corresponds to the given action $\alpha_i$.

If the penalty probabilities vary with time, the environment is called a non-stationary environment and it is said to be stationary otherwise. Depending on the nature of the reinforcement signal $\beta$, random environments are categorized into $P$-model, $Q$-model and $S$-model. In $P$-model environments, $\beta$ can only take two binary values $0$ and $1$. In contrast, $Q$-model environments allow the signal $\beta$ to take a finite number of the values in the interval $[0,1]$. Finally, in S-model environments, $\beta$ lies in the interval $[a, b]$.

Stochastic learning automata can be categorized into two main classes: variable structure learning automata (VSLA), in which the transition function and output function vary over time, and otherwise fixed structure learning automata (FSLA). VSLA can be represented by a quadruple $\langle \alpha, \beta, p, T \rangle$, where:

- $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ indicates the action set from which the automaton chooses.
- $\beta = \{\beta_1, \beta_2, \dots, \beta_k\}$ indicates the set of inputs to the automaton.
- $p = \{p_1, p_2, \dots, p_r\}$ indicates the action probability vector, such that $p_i$ is the probability of selecting the action $\alpha_i$.
- $T$ indicates the learning algorithm that is used to update the action probability vector of the automaton in terms of the environment's response, i.e. $p(t+1) = T[\alpha(t), \beta(t), p(t)]$, where the inputs are the chosen action $\alpha(t)$, the response of the environment $\beta(t)$ and the action probability vector $p(t)$ at time $t$.

Let $\alpha_i(t)$ be the action selected by the automaton at time $t$. The action probability vector $p(t)$ is updated as given in Eq. 1, if the environment's response is reward, and $p(t)$ is updated according to Eq. 2, if the response of the environment is penalty.

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

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

where $r$ is the number of actions available for the automaton, and $a$ and $b$ are respectively reward and penalty parameters determining the amount of increases and decreases in the action probabilities. If $a = b$, the learning algorithm $T$ is a reward-penalty ($L_{R-P}$) algorithm, if $a \gg b$, it is a reward-$\epsilon$ penalty ($L_{R-\epsilon P}$) algorithm, and if $b = 0$, it is a reward-Inaction ($L_{R-I}$) algorithm in which the action probability vector $p(t)$ remains unchanged when the response of the environment is penalty. Stochastic learning automaton has been widely used in literature for solving various problems, for example, graph sampling [49,50], fuzzy membership function optimization [51], trust propagation [9], Multicast Routing [52].

A variable structure learning automaton in which the number of available actions varies in time is called a variable action set learning automaton. Assume that $\hat{\alpha}(t) \subseteq \alpha$ is an action subset available for the automaton at time $t$, where the elements of $\hat{\alpha}(t)$ are randomly selected by an external factor. Also let $K(t) = \sum_{\alpha_i \in \hat{\alpha}(t)} p_i(t)$ be the sum of the probability of the available actions in $\hat{\alpha}(t)$. Before the selection of an action, the available actions probability vector $\hat{p}(t)$ is scaled by Eq. 3.

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

Afterward, the automaton randomly selects one of its available actions from $\hat{\alpha}(t)$ according to the scaled action probability vector $\hat{p}(t) = \{\hat{p}_i(t) | \alpha_i \in \hat{\alpha}(t)\}$. Depending on the response received from the environment, the automaton updates $\hat{p}(t)$. Finally, the available actions probability vector $\hat{p}(t)$ is rescaled as given in Eq. 4.

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

In order to investigate the learning ability of a learning automaton, a pure-chance automaton that always selects its available actions with equal probabilities is used as the standard for comparison [48]. Any automaton that is said to learn must perform at least better than such a pure-chance automaton. To make this comparison, one measure can be the average penalty for a given action probability vector. For a stationary random environment with the penalty probability vector $c = \{c_1, c_2, \dots, c_r\}$, the average penalty probability $M(t)$ received by an automaton is equal to

$$M(t) = E[\beta(t) | p(t)] = \sum_{\alpha_i \in \alpha} c_i p_i(t) \tag{5}$$

For a pure-chance automaton, $M(t)$ is a constant and is defined as

$$M(0) = \frac{1}{r} \sum_{\alpha_i \in \alpha} c_i \tag{6}$$

An automaton that does better than pure chance must have the average penalty $M(t)$ less than $M(0)$ at least asymptotically as $t \to \infty$. Since $p(t)$ and consequently $M(t)$ are random variables in general, the expected value $E[M(t)]$ is compared with $M(0)$.

**Definition 1.** A learning automaton interacting with a $P$-, $Q$-, or $S$-model environment is said to be *expedient* if

$$\lim_{t \to \infty} E[M(t)] < M(0) \tag{7}$$

Expediency means that the average penalty probability decreases when the automaton updates its action probability function. It would be more interested to determine an updating procedure which would result in $E[M(t)]$ attaining its minimum value. In such a case, the automaton is called *optimal*.

**Definition 2.** A learning automaton interacting with a $P$-, $Q$-, or $S$-model environment is said to be *optimal*

$$\lim_{t\to\infty} E[M(t)] = c_\ell \tag{8}$$

where $c_\ell = \min_i c_i$. Optimality implies that asymptotically the action $\alpha_\ell$ with the minimum penalty probability $c_\ell$ is chosen with probability one. While optimality is a very desirable property in stationary environments, it may not be possible to achieve it in a given situation. In such case, one might aim at a suboptimal performance which is represented by $\varepsilon$-*optimality*.

**Definition 3.** A learning automaton interacting with a $P$-, $Q$-, or $S$-model environment is said to be $\varepsilon$-*optimal* if the following equation can be obtained for any $\varepsilon > 0$ by a proper choice of the parameters of the learning automaton.

$$\lim_{t\to\infty} E[M(t)] < c_\ell + \varepsilon \tag{9}$$

$\varepsilon$-optimality implies that the performance of the automaton can be made as close to the optimal as desired.

## 3.2 Distributed Learning Automata

Distributed learning automata (DLA) [53] is a network of interconnected automata which cooperate together to solve a particular problem. Formally, DLA can be represented by a quadruple $\langle A, E, \mathcal{T}, A_0\rangle$, where

- $A = \{A_1, A_2, \dots, A_n\}$ is a set of learning automata.
- $E \subset A \times A$ denotes an edge set (where edge $e_{ij}$ corresponds to the action $\alpha_{ij}$ of the learning automaton $A_i$).
- $\mathcal{T}$ denotes a set of learning algorithms based on which the automata update their action probability vectors.
- $A_0$ is the root automaton from which the activation process of the automata begins.

DLA has exactly one root automaton which is always the first to be activated, and at least one leaf automaton (an automaton which interacts with the random environment) which is activated probabilistically. Figure 1 depicts an example DLA.

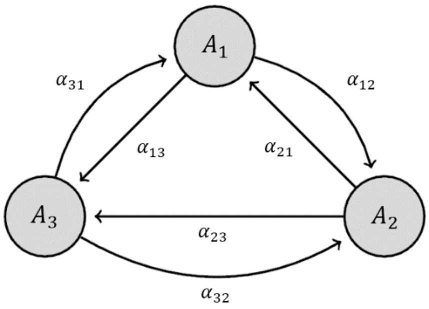


**Fig. 1.** Distributed learning automata

A distributed learning automata operates as follows. At first, the root automaton is activated and randomly selects one of its outgoing edges (i.e. actions) according to the corresponding action probabilities. As a result of this action selection, the automaton on the other end of the selected edge will be activated. The activated automaton also selects randomly one of its actions which results in the activation of another automaton. The process of choosing an action and activating another automaton is continued until a leaf automaton is reached. The selected actions along the path induced by the activation process of the automata are applied to the environment. Depending on the response received from the environment, the activated learning automata along the path update their action probability vectors. Traveling different paths from the root automaton to the leaf automata is continued until the choice probability of a path is close enough to unity.

# 4. PROPOSED DYNAMIC ALGORITHM FOR STOCHASTIC TRUST PROPAGATION

## 4.1 Notations and Definitions

We first present some notations and definitions below that are used to describe our proposed algorithm.

**Definition 4 (Stochastic trust network).** A trust network can be modelled by a weighted digraph $G(V,E,W)$, where $V = \{v_1, v_2, \dots, v_n\}$ is the set of $n$ users, $E = \{e_{ij} | \; v_i,v_j \in V\}$ is the set of $m$ directed trust relations such that $e_{ij}$ refers to the trust relation from $v_i$ to $v_j$, and $W = \{w_{ij} | \; e_{ij} \in E\}$ is the set of trust weights where $w_{ij}$ denotes the weight associated with the trust relation $e_{ij}$. Considering the dynamic nature of trust, $G$ is a stochastic network, in which each trust weight $w_{ij}$ is a random variable.

It is assumed that the weights of trust relations are independent random variables with stationary distributions. We also assume that each random variable $w_{ij}$ takes real values in interval [0,1], with 1 referring to full trust and 0 to no trust.

**Definition 5 (Stochastic trust propagation).** Given a stochastic trust network $G(V,E,W)$, a source user $v_s$ and an indirectly connected target user $v_d$. The goal of a Stochastic trust propagation problem is to infer the trust value of $v_s$ related to $v_d$ via propagating trust along reliable trust paths between the two users in the stochastic network $G$, while capturing the dynamicity of trust weights during the trust propagation process and dynamically updating the found reliable paths upon the trust variations.

Let $\mathcal{N}_d$ denote the set of users directly trusting the target $v_d$. Hereinafter, we refer to the users in $\mathcal{N}_d$ as the direct neighbors of $v_d$ and to their trust statements as the direct trust values. The proposed algorithm in this paper utilizes learning automata to capture the changes of trust weights and find the most reliable trust path to each neighboring node $v_{n_k} \in \mathcal{N}_d$. Since the weight of each trust relation is a random variable, the algorithm uses the fundamental technique of sampling the trust weights, as an approach to uncertainty in stochastic trust network, and determines the most reliable path with respect to the expected values of trust weights.

## 4.2 Algorithm Description

In this section, we describe our dynamic algorithm for propagating stochastic trust, called DyTrust. The inputs to the algorithm DyTrust are a stochastic trust network $G(V, E, W)$, a source node $v_s$ and an indirectly connected target node $v_d$, and the output is the estimated trust weight $w'_{sd}$ of $v_s$ on $v_d$. The proposed algorithm uses a distributed learning automata (DLA) to discover the most reliable trust path to each direct neighbor of $v_d$ based on samples taken from trust weights during its running time. The stochastic trust network $G$ plays the role of random environment for DLA. One advantage of our proposed algorithm is that it needs no knowledge about probability distributions of trust weights.

The pseudo code for DyTrust is given in Fig. 2 and can be described as follows. At first, a network of learning automata is formed by assigning an automaton $A_i$ to each node $v_i \in V$ of $G$. That is, two notations $A_i$ and $v_i$ refer to the same concept and so might be used interchangeably. For each automaton $A_i$, the action set $\alpha_i$ includes all the nodes directly trusted by $v_i$ (or outgoing trust links), and the learning algorithm $L_{R-I}$ is used for updating the action probability vector of the automaton. The automaton $A_s$ corresponding to the source $v_s$ is considered as the root automaton from which the activation process of the learning automata always starts.

After constructing a DLA isomorphic to the input stochastic trust network $G$, for each direct neighbor $v_{n_k}$ of the target $v_d$ (i.e. for each $v_{n_k} \in \mathcal{N}_d$) the algorithm DyTrust first initializes the action probability vectors of all the learning automata and then finds the most reliable trust path to $v_{n_k}$. The action probability vector $p_i$ of each automaton $A_i$ is initialized according to Eq. 10.

$$p_{ij}(t) = \frac{\tau_{ij}(t-1)}{\sum_{v_k \in \alpha_i} \tau_{ik}(t-1)} \quad \forall v_j \in \alpha_i \tag{10}$$

where $p_{ij}(t)$ is the selection probability of node $v_j$ by $A_i$ at time $t$ and $\tau_{ij}(t-1)$ denotes the trust value of node $v_i$ on $v_j$ at time $t-1$ which is computed as given in Eq. 11. Using the above equation for initializing the action probability vectors of the learning automata, highly trusted nodes are initially assigned higher probabilities of selection.

$$\tau_{ij}(t) = \frac{\sum_{l=1}^{|\omega_{ij}(t)|} \lambda^{|\omega_{ij}(t)|-l}\, \omega_{ij}^{l}(t)}{\sum_{l=1}^{|\omega_{ij}(t)|} \lambda^{|\omega_{ij}(t)|-l}} \tag{11}$$

where $\omega_{ij}(t)$ is the set of trust weights observed on $e_{ij}$ until current time $t$, and so $\omega_{ij}^{l}(t)$ refers to $l$th member and $|\omega_{ij}(t)|$ denotes the size of this set. $\lambda \in [0,1]$ is a decay factor that controls the rate at which old trust weights are aged and discounted.

In order to discover the most reliable path to the direct neighbor $v_{n_k}$, DyTrust repeats the following three steps until the stopping condition is reached.

*Step 1. Discovering a trust path towards direct neighbor*

In this step, a trust path $\pi$ from the source $v_s$ towards the direct neighbor $v_{n_k}$ of $v_d$ will be found by a chain of automaton activations. For this purpose, at first the root automaton $A_s$ is activated to choose one of its actions, say action $v_i$, based on the corresponding action probability distribution. As a result of this action selection, a sample is taken from link $e_{si}$, the trust value $\tau_{si}$ of the link is recomputed by Eq. 11 and the automaton $A_i$ on the other end of $e_{si}$ is activated as the next hop along the path $\pi$. If the activated automaton $A_i$ belongs to the set $\mathcal{N}_d$ and the minimum of trust values along $\pi$, called the path strength $\mathcal{S}_\pi$, is larger than or equal to the maximum strength $\mathcal{S}_i$ already obtained for $v_i$, $\mathcal{S}_i$ is updated to the strength of the path $\pi$, namely if $\mathcal{S}_\pi = \min_{e_{ij} \in \pi} \left(\tau_{ij}(t)\right) \geq \mathcal{S}_i$ then $\mathcal{S}_i = \mathcal{S}_\pi$. After that, the set of available actions for $A_i$

is determined as all the neighbors directly trusted by $v_i$ except those whose corresponding automaton has been already activated along the current path. The automaton $A_i$ chooses one of its available actions based on the scaled action probability distribution. The process of choosing an action and activating another automaton is repeated until the direct neighbor $v_{n_k}$ is reached or there is no available action for the activated automaton.

```
**Algorithm** DyTrust : Dynamic algorithm for trust propagation
**Input**   Stochastic trust network $G(V, E, W)$ , Source node $v_s$, Target node $v_d$
**Output**   Estimated trust $w'_{sd}$
**Begin**
    Assign an automaton $A_i$ to each node $v_i$ of $G$
    Let $\mathcal{N}_d$ be the set of target's trusting neighbors
    Set the reliability $\mathcal{S}_i$ of each node $v_i \in \mathcal{N}_d$ to 0
    Let $\omega_{ij}$ be the set of trust weights observed on each trust link $e_{ij}$ and is initially set to empty
    $t = 0$
    **For** each direct neighbor $v_{n_k} \in \mathcal{N}_d$ **Do**
        Initialize the action probability vector $p_i(t)$ of each automaton $A_i$ by Eq. 10
        Let $\rho$ be the number of found paths and is initially set to 0
        **Repeat**
            $A_u \leftarrow$ automaton $A_s$ corresponding to the source node $v_s$
            Let $\pi$ be the set of travelled edges along path from $v_s$ towards $v_{n_k}$ and is initially set to empty
            **Repeat**
                Let the set of available actions for $A_u$ include all $v_u$'s trusted neighbors except traversed ones
                **If** the available action set for $A_u$ is not empty **then**
                    $v_i \leftarrow A_u$ chooses one of its available actions according to $\hat{p}_u(t)$
                    Take a sample from link $e_{ui}$ and update the trust value $\tau_{ui}$ of $e_{ui}$ by Eq. 11
                    $\pi \leftarrow \pi \cup e_{ui}$
                    $A_u \leftarrow A_i$
                    Let $\mathcal{S}_\pi$ be the strength of path $\pi$ from $A_s$ to $A_u$
                    **If** $v_u \in \mathcal{N}_d$ **and** $\mathcal{S}_\pi \geq \mathcal{S}_u$ **then**
                        $\mathcal{S}_u \leftarrow \mathcal{S}_\pi$
                    **End**
                **End**
            **Until** ( There is no available action **or** $v_u = v_{n_k}$ )
            **If** $v_u = v_{n_k}$ **and** $\mathcal{S}_\pi \geq \mathcal{S}_{n_k}$ **then**
                Reward the activated learning automata along path $\pi$
            **End**
            $\rho \leftarrow \rho + 1$
            $t \leftarrow t + 1$
        **Until** (the probability of path $\pi > \mathcal{P}$ **or** $\rho > K$ )
    **End**
    Compute the estimated trust $w'_{sd}$ by Eq. 13
**End Algorithm**
```

**Fig. 2.** The pseudo code of DyTrust algorithm

*Step 2. Updating action probability vectors*

If in the previous step the found path $\pi$ ends in the direct neighbor $v_{n_k}$ and the path strength $\mathcal{S}_\pi$ is larger than or equal to the maximum strength $\mathcal{S}_{n_k}$, the activated learning automata along the trust path $\pi$ will be rewarded. The value of the reward parameter $a$ for each automaton $A_j$ along the path is determined based on Eq. 12 [53].

$$a_j(t) = \frac{a_i(t)}{p_{ij}(t+1)} \quad \forall e_{ij} \in \pi \tag{12}$$

where $a_i(t)$ denotes the reward parameter of automaton $A_i$ activated along the path $\pi$ at time $t$ and $p_{ij}(t+1)$ is the probability of being chosen node $v_j$ by $A_i$ after rewarding the automaton.

*Step 3. Checking stop condition*

The process of finding a trust path and evaluating the path continues until the product of the selection probability of the trust links along the path, called path probability, is greater than a predefined threshold $\mathcal{P}$ or the number of traversed paths exceeds a certain threshold $K$. In this situation, the maximum strength $\mathcal{S}_{n_k}$ obtained for the direct neighbor $v_{n_k}$ is considered as the reliability of the direct trust issued by the neighbor on the target $v_d$.

After measuring the reliability of the target's direct neighbors, the direct trust values are combined to infer trust of the source on the target node. For this purpose, we use the aggregation function proposed in our previous work [9], called MCFAvg, as given below.

$$w'_{sd} = \bar{\tau}_s + \frac{\sum_{v_i \in \mathcal{N}_d} \mathcal{S}_{\boldsymbol{i}} \left(\tau_{id} - \bar{\tau}_i\right)}{|\mathcal{N}_d|\; Max_w} \tag{13}$$

where $|\mathcal{N}_d|$ denotes the size of set $\mathcal{N}_d$, $Max_w$ is the maximum trust weight which is equal to 1 in this paper, and $\bar{\tau}_i$ is the average of the trust values provided by $v_i$ on its trusted neighbors.

## 5. CONVERGENCE RESULTS

This section aims to prove the convergence of our proposed algorithm DyTrust based on the assumption that trust weight distributions in stochastic trust network are stationary, i.e., their statistical properties do not change over time. Since DyTrust infers the trust value of source user on given target based on aggregating the opinions of the target's direct neighbors weighted by their reliabilities, it will be enough to prove that DyTrust converges to the most reliable trust path to each direct neighbor with a probability as close to unity as desired. We do this in two steps. The first step concerns with the proof of the convergence of DyTrust to the trust path with the highest expected reliability (the optimal solution) by the proper choice of the learning parameter $a$ when the learning algorithm of each automaton is the algorithm $L_{R-I}$. That is, this step indicates that by choosing a proper learning parameter, the choice probability of the optimal solution in DyTrust converges to one as much as possible. The second step determines a learning parameter under which the probability of finding (or converging to) the optimal solution exceeds $1 - \varepsilon$. That is, this step concerns with the relationship between the convergence error parameter $\varepsilon$ and the learning parameter $a$ of DyTrust.

**Theorem 1.** Let $q_i(t)$ be the probability of traversing along trust path $\pi_i$ in a stochastic trust network at time $t$ and $\underline{q}(t) = \left(q_1(t), q_2(t), \dots, q_\rho(t)\right)^T$, where $\rho$ is the number of all trust paths between source and direct neighbor in the network. If $\underline{q}(t)$ is updated according to the algorithm DyTrust, then for every $\varepsilon > 0$, there exists a learning parameter $a^* \in (0,1)$ such that for all $a \in (0, a^*)$, we have

$$Prob\left[\lim_{t\to\infty} q_l(t) = 1\right] \geq 1 - \varepsilon \tag{14}$$

where $q_l(t)$ denotes the probability of choosing the most reliable trust path $\pi^*$. This theorem asserts that DyTrust converges to the most reliable path with a probability as close to unity as desired.

*Proof.* The method used for proving the above theorem is similar to the method given in [48,54,55] for analyzing the behavior of learning automaton in non-stationary environments. The convergence proof is done in the following three steps. At first, it is shown in Lemma 1 that for large enough value of $t$, the penalty probability of each trust path converges to the constant value of the final penalty probability. Then, in Lemmas 2 and 3, it is proved that the probability of choosing the trust path with the highest expected reliability is a sub-Martingale process for large values of $t$, and so the changes in the probability of travelling the most reliable trust path is always non-negative. Finally, the convergence of DyTrust to the most reliable path is proved using Martingale convergence theorems. □

**Lemma 1.** Assume that the trust path $\pi_i$ is penalized with probability $c_i(t)$ at time $t$ and let $c_i(t) = Prob[\mathcal{S}_{\pi_i}(t) < \mathcal{S}_{n_k}(t)]$ and $\lim_{t\to\infty} c_i(t) = c_i^*$. Then, for each $\varepsilon \in (0,1)$ and $t > t_\varepsilon$ (the minimum time needed by DyTrust to achieve the error rate $\varepsilon$), we have

$$Prob[|c_i^* - c_i(t)| > 0] < \varepsilon \tag{15}$$

*Proof.* Let $c_i^*$ be the final value of the penalty probability $c_i(t)$ when $t$ is large enough. That is, for large enough values of $t$, the probability of penalizing the trust path $\pi_i$ (i.e. $c_i(t)$) converges to its true value $c_i^*$. Using weak law of large numbers, we conclude that

$$\lim_{t\to\infty} Prob[|c_i^* - c_i(t)| > \varepsilon] \to 0$$

Hence, for any $\varepsilon \in (0,1)$, there exists a learning parameter $a_\varepsilon^* \in (0,1)$ and $t_\varepsilon < \infty$ such that for all $a < a_\varepsilon^*$ and $t > t_\varepsilon$, we have $Prob[|c_i^* - c_i(t)| > 0] < \varepsilon$, and the proof of the lemma is completed. □

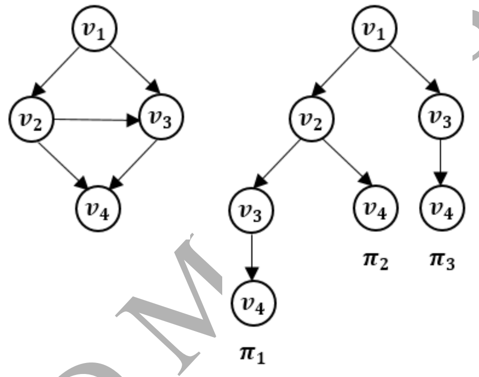
**Lemma 2.** Let $c_i(t) = Prob[\mathcal{S}_{\pi_i}(t) < \mathcal{S}_{n_k}(t)]$ and $d_i = 1 - c_i$ be respectively the probability of penalizing and rewarding the trust path $\pi_i$. If $\underline{q}(t)$ is updated according to the algorithm DyTrust, then the conditional expectation of $q_i(t)$ is given as

$$E[q_i(t+1)|q(t)] = \sum_{j=1}^{\rho} q_j(t)\left[c_j(t)q_i(t) + d_j(t)\prod_{e_{uv}\in\pi_i} p_{uv}(t+1)\right] \tag{16}$$

where $\rho$ denotes the number of all trust paths from source to direct neighbor and $p_{uv}(t+1)$ is computed as

$$p_{uv}(t+1) = \begin{cases} p_{uv}(t) + a[1 - p_{uv}(t)] & e_{uv} \in \pi_j \\ (1-a)p_{uv}(t) & e_{uv} \notin \pi_j \end{cases}$$

*Proof.* As stated before, the probability of choosing a trust path $\pi_i$ is computed as the product of the selection probability of the trust links along the path, i.e. $q_i(t) = \prod_{e_{uv} \in \pi_i} p_{uv}(t)$. Since DyTrust uses the learning algorithm $L_{R-I}$ to update the action probability vectors at each time $t$, the probability $q_i(t)$ of traversing the trust path $\pi_i$ remains unchanged with the probability $c_j(t)$, if the chosen trust path $\pi_j$ is penalized by the environment, and otherwise, if the path $\pi_j$ is rewarded with the probability $d_j(t)$, the probability of choosing the edges of the trust path $\pi_i$ which are in the selected path $\pi_j$ increases by a given learning rate as that of the other edges of $\pi_i$ decreases. To illustrate the proof of the lemma in more detail, we prove it for the optimal trust path of the trust network given in Figure 3(a). As shown in Figure 3(b), there exist three trust paths from node $v_1$ to target's neighbor $v_4$ including: $\pi_1 = \{e_{12}, e_{23}, e_{34}\}$, $\pi_2 = \{e_{12}, e_{24}\}$ and $\pi_3 = \{e_{13}, e_{34}\}$. It is assumed that $\pi_2$ is the optimal trust path from $v_1$ to $v_4$.



a) b)

**Fig. 3.** a) Sample trust network $G$ and b) its search tree

Let $q_i(t)$ be the probability of traversing the trust path $\pi_i$ and $p_{uv}$ the probability of choosing action $\alpha_{uv}$ by the automaton $A_u$, at time $t$. Therefore, we have

$$q_1(t) = p_{12}(t)p_{23}(t)p_{34}(t)$$
$$q_2(t) = p_{12}(t)p_{24}(t)$$
$$q_3(t) = p_{13}(t)p_{34}(t)$$

The conditional expectation of $q_2(t+1)$, assuming $\underline{q}(t)$ is updated according to the algorithm DyTrust, is computed as

$$E[q_2(t+1)|q(t)] =$$
$$q_1(t)[c_1(t)q_2(t) + d_1(t)\{p_{12}(t) + a(1-p_{12}(t))\}\{(1-a_2(t))p_{24}(t)\}] +$$
$$q_2(t)[c_2(t)q_2(t) + d_2(t)\{p_{12}(t) + \mathrm{a}(1-p_{12}(t))\}\{p_{24}(t) + \mathrm{a}_2(\mathrm{t})(1-p_{24}(t))\}] +$$
$$q_3(t)[c_3(t)q_2(t) + d_3(t)\{(1-\mathrm{a})p_{12}(t)\}p_{24}(t)]$$

where $a_u(t)$ is the step length of the automaton $A_u$ at time $t$ and is computed according to Eq. 12. After simplifying all terms in the right side of the above equation and some algebraic manipulations, we have

$$E[q_2(t+1)|q(t)] = \sum_{j=1}^{3} q_j(t)\left[c_j(t)q_2(t) + d_j(t)\prod_{e_{uv}\in\pi_2} p_{uv}(t+1)\right]$$

Hence, the proof of the lemma is completed. □

**Lemma 3.** If $\pi_l$ is the trust path with the highest expected reliability and $\underline{q}(t)$ is updated according to the algorithm DyTrust, the increment in the conditional expectation of $q_l(t)$ is always non-negative, that is, $\Delta q_l(t) = E[q_l(t+1)|q(t)] - q_l(t) > 0$ for all $q_l(t) \in (0,1)$.

*Proof.* From lemma 2, we have

$$\Delta q_l(t) = E[q_l(t+1)|q(t)] - q_l(t) = \sum_{j=1}^{\rho} q_j(t)\left[c_j(t)q_l(t) + d_j(t)\prod_{e_{uv}\in\pi_l} p_{uv}(t+1)\right] - q_l(t) \tag{17}$$

Since The probability of traversing, rewarding, and penalizing a trust path is defined as the product of the probability of choosing the trust links along the path, the above equality can be rewritten as

$$\Delta q_l(t) = \sum_{j=1}^{\rho}\prod_{e_{uv}\in\pi_j} p_{uv}(t)\left[\prod_{e_{uv}\in\pi_j} c_{uv}(t)\prod_{e_{uv}\in\pi_l} p_{uv}(t) + \prod_{e_{uv}\in\pi_j} d_{uv}(t)\prod_{e_{uv}\in\pi_l} p_{uv}(t+1)\right] - \prod_{e_{uv}\in\pi_l} p_{uv}(t)$$

from which we have

$$\Delta q_l(t) = \prod_{e_{uv}\in\pi_l} E[p_{uv}(t+1)|p_u(t)] - \prod_{e_{uv}\in\pi_l} p_{uv}(t)$$

This equality can be rewritten as

$$\Delta q_l(t) \geq \prod_{e_{uv}\in\pi_l}\left(E[p_{uv}(t+1)|p_u(t)] - p_{uv}(t)\right) = \prod_{e_{uv}\in\pi_l} \Delta p_{uv}(t) \tag{18}$$

and

$$\Delta p_{uv}(t) = ap_{uv}(t)\sum_{v_h\in\alpha_u, h\neq v} p_{uh}(t)\left(c_{uh}(t) - c_{uv}(t)\right)$$

$q_i(t) \in (0,1)$ for all $\underline{q} \in S_\rho^0$, where $S_\rho = \{\underline{q}(t): 0 \leq q_i(t) \leq 1; \sum_{i=1}^{\rho} q_i(t) = 1\}$ and $S_\rho^0$ is the interior of $S_\rho$. Hence, $p_{uv}(t) \in (0,1)$ for all $u, v$. Since the trust path $\pi_l$ is the path with the highest expected reliability, then $c_j^* - c_l^* > 0$ for all $j \neq l$ and hence for each $e_{uv} \in \pi_l$, $c_{uh}^* - c_{uv}^* > 0$ for any action $v_h$ $(h \neq v)$ of automaton $A_u$. It follows form lemma 1 that $c_{uh}(t) - c_{uv}(t) > 0$ for large values of $t$. Therefore, we can conclude that for large values of $t$, the right side of the above equation is positive and so considering Equation 18, we have

$$\Delta q_l(t) \geq \prod_{e_{uv}\in\pi_l} ap_{uv}(t) \sum_{v_h\in\alpha_u, h\neq v} p_{uh}(t)\big(c_{uh}(t) - c_{uv}(t)\big) \geq 0$$

which completes the proof of this lemma. □

**Corollary 1.** The set of unit vectors in $S_\rho - S_\rho^0$, where $S_\rho^0 = \{\underline{q}(t) : q_i(t) \in (0,1); \sum_{i=1}^{\rho} q_i(t) = 1\}$, constitutes the set of all absorbing barriers of the Markov process $\left\{\underline{q}(t)\right\}_{t\geq 1}$.

*Proof.* Lemma 3 implicitly implies that $\{q(t)\}$ is a sub-Martingale. Considering Martingale theorems and the fact that $\{q(t)\}$ is non-negative and uniformly bounded, it can be concluded that $\lim_{t\to\infty} q_l(t)$ converges to $q^*$ with probability one. Further, from Equation 17, it is observed that if $q_l(t) \notin \{0,1\}$, then $q_l(t+1) \neq q_l(t)$ with a nonzero probability for all $t$, and if $q^* \in \{0,1\}$, then $\underline{q}(t+1) = \underline{q}(t)$, hence the proof of this lemma. □

Let $\Gamma_l(q)$ be the convergence probability of Dytrust to the unit vector $I_l$ with initial probability vector $\underline{q}$, which is defined as

$$\Gamma_l(q) = Prob\left[q_l(\infty) = 1 \middle| \underline{q}(0) = \underline{q}\right] = Prob[q^* = I_l | \underline{q}(0) = \underline{q}] \tag{19}$$

Also let $C(S_\rho): S_\rho \to \Re$, where $\Re$ is the real line, be the state space of all real-valued continuously differentiable functions with bounded derivative defined on $S_\rho$, and $\psi(.) \in C(S_\rho)$. The operator $U$ is defined as follows

$$U\psi(q) = E[\psi(q(t+1)) | q(t) = q] \tag{20}$$

Work in [55] has been shown that the operator $U$ is linear and preserves the non-negative functions as the expectation of a positive function remains positive. In other word, if $\psi(q) \geq 0$, then $U\psi(q) \geq 0$ for all $q \in S_\rho$. Function $\psi(q)$ is called sub-regular (super-regular) if and only if $\psi(q) \leq U\psi(q)$ $(\psi(q) \leq U\psi(q))$ for all $q \in S_\rho$. It has also been shown in [55] that $\Gamma_l(q)$ is the only continuous solution of $U\Gamma_l(q) = \Gamma_l(q)$, subject to the following boundary conditions.

$$\Gamma_l(I_j) = \begin{cases} 1 & j = l \\ 0 & j \neq l \end{cases} \tag{21}$$

$\phi_l[x, q] \in C(S_\rho)$, defined as given below, satisfies the above boundary conditions.

$$\phi_l[x, q] = \frac{e^{-xq_l/a} - 1}{e^{-x/a} - 1} \tag{22}$$

where $x > 0$.

**Theorem 2.** If $\psi_l(.) \in C(S_\rho)$ be super-regular with the boundary conditions $\psi_l(I_l) = 1$ and $\psi_l(I_j) = 0$ for $j \neq l$, then $\psi_l(q) \geq \Gamma_l(q)$ for all $q \in S_\rho$. Also if $\psi_l(.) \in C(S_\rho)$ be sub-regular with the same boundary conditions, then $\psi_l(q) \leq \Gamma_l(q)$ for all $q \in S_\rho$.

*Proof.* This theorem has been proved in [48] . □

In what follows, it is shown that $\phi_l[x,q]$ is a sub-regular function and thus $\phi_l[x,q]$ qualifies as a lower bound on $\Gamma_l(q)$. Since sub and super-regular functions are closed under addition and multiplication by a positive constant, and if $\phi(.)$ is a super-regular function then $-\phi(.)$ is sub-regular, $\phi_l[x,q]$ is sub-regular if and only if $\theta_l[x,q] = e^{-xq_l/a}$ is super-regular. The conditions under which $\theta_l[x,q]$ is super-regular are determined as follows. From the definition of the operator $U$ in Equation 20, we have

$$\begin{aligned} U\theta_l[x,q] &= E\left[e^{-xq_l(t+1)/a} | q(t) = q\right] \\ &= \sum_{j=1}^{\rho} q_j d_j^* e^{\frac{-x}{a}\left[\prod_{\substack{e_{uv}\in\pi_l, \\ e_{uv}\in\pi_j}}(p_{uv}+a(1-p_{uv}))\right]} + \sum_{j=1}^{\rho} q_j d_j^* e^{\frac{-x}{a}\left[\prod_{\substack{e_{uv}\in\pi_l, \\ e_{uv}\notin\pi_j}}(p_{uv}(1-a))\right]} \\ &= q_l d_l^* e^{\frac{-x}{a}(q_l+a(1-q_l))} + \sum_{j\neq l} q_j d_j^* e^{\frac{-x}{a}\left[\prod_{\substack{e_{uv}\in\pi_l, \\ e_{uv}\in\pi_j}}(p_{uv}+a(1-p_{uv}))\right]} \\ &+ \sum_{j\neq l} q_j d_j^* e^{\frac{-x}{a}\left[\prod_{\substack{e_{uv}\in\pi_l, \\ e_{uv}\notin\pi_j}}(p_{uv}(1-a))\right]} \end{aligned} \tag{23}$$

where $d_j^*$ is the final value to which the reward probability $d_j(t)$ converges for large values of $t$, and $e^{\frac{-x}{a}(q_l+a(1-q_l))}$ denotes the expectation of $\theta_l[x,q]$ when the most reliable path $\pi_l$ is rewarded by the environment.

$$\begin{aligned} U\theta_l[x,q] &= q_l d_l^* e^{\frac{-x}{a}(q_l+a(1-q_l))} + \sum_{j\neq l} q_j d_j^* e^{\frac{-x}{a}\left[q_l(1-a)\prod_{\substack{e_{uv}\in\pi_l, \\ e_{uv}\in\pi_j}}\left(\frac{p_{uv}+a(1-p_{uv})}{p_{uv}(1-a)}\right)\right]} \\ &= \sum_{j=l} q_j d_j^* e^{\frac{-x}{a}\delta_{lj}(q_l+a(1-q_l))} + \sum_{j\neq l} q_j d_j^* e^{\frac{-x}{a}(\delta_{lj} q_l(1-a))} \end{aligned}$$

where $\delta_{lj}$ is computed as

$$\delta_{lj} = \begin{cases} \prod\limits_{\substack{e_{uv}\in\pi_l, \\ e_{uv}\in\pi_j}} \left(\dfrac{p_{uv}+a(1-p_{uv})}{p_{uv}(1-a)}\right) & j \neq l \\ 1 & j = l \;\; or \;\; (\pi_j \cap \pi_l) = \emptyset \end{cases}$$

$$U\theta_l[x,q] - \theta_l[x,q] = \left[e^{\frac{-xq_l\delta_{lj}}{a}} \sum_{j=l} q_j d_j^* e^{-x(1-q_l)\delta_{lj}} + e^{\frac{-xq_l\delta_{lj}}{a}} \sum_{j\neq l} q_j d_j^* e^{-xq_l\delta_{lj}}\right] - e^{\frac{-xq_l}{a}}$$

$\theta_l[x,q]$ is super-regular if

$$e^{\frac{-xq_l\delta_{lj}}{a}} \sum_{j=l} q_j d_j^* e^{-x(1-q_l)\delta_{lj}} + e^{\frac{-xq_l\delta_{lj}}{a}} \sum_{j\neq l} q_j d_j^* e^{-xq_l\delta_{lj}} \leq e^{\frac{-xq_l}{a}}$$

and

$$U\theta_l[x,q] \leq e^{\frac{-xq_l}{a}} q_l d_l^* e^{-x(1-q_l)} + e^{\frac{-xq_l}{a}} \sum_{j\neq l} q_j d_j^* e^{-xq_l}$$

If $\theta_l[x,q]$ is super-regular, then we have

$$U\theta_l[x,q] - \theta_l[x,q] \leq \left[ e^{\frac{-xq_l}{a}} q_l d_l^* e^{-x(1-q_l)} + e^{\frac{-xq_l}{a}} \sum_{j\neq l} q_j d_j^* e^{-xq_l} \right] - e^{\frac{-xq_l}{a}}$$

After multiplying and dividing the right side of the above inequality by $-xq_l$ and some algebraic simplifications, we have

$$\begin{aligned} U\theta_l[x,q] - \theta_l[x,q] &\leq -xq_l e^{\frac{-xq_l}{a}} \left[ q_l d_l^* \frac{e^{-x(1-q_l)} - 1}{-xq_l} - \sum_{j\neq l} q_j d_j^* \frac{e^{xq_l} - 1}{xq_l} \right] \\ &= -xq_l e^{\frac{-xq_l}{a}} \left[ d_l^* \frac{e^{-x(1-q_l)} - 1}{-x} - \sum_{j\neq l} q_j d_j^* \frac{e^{xq_l} - 1}{xq_l} \right] \\ &= -xq_l e^{\frac{-xq_l}{a}} \left[ (1-q_l) d_l^* \frac{e^{-x(1-q_l)} - 1}{-x(1-q_l)} - \sum_{j\neq l} q_j d_j^* \frac{e^{xq_l} - 1}{xq_l} \right] \end{aligned}$$

and

$$\begin{aligned} U\theta_l[x,q] - \theta_l[x,q] &\leq -xq_l e^{\frac{-xq_l}{a}} \left[ (1-q_l) d_l^* V[-x(1-q_l)] - \left( \sum_{j\neq l} q_j d_j^* \right) V[xq_l] \right] \\ &= -xq_l \theta_l[x,q] G_l[x,q] \end{aligned}$$

where

$$V[u] = \begin{cases} \dfrac{e^u - 1}{u} & u \neq 0 \\ 1 & u = 0 \end{cases}$$

and

$$G_l[x,q] = (1-q_l) d_l^* V[-x(1-q_l)] - \left( \sum_{j\neq l} q_j d_j^* \right) V[xq_l] \tag{24}$$

Therefore, $\theta_l[x,q]$ is super-regular if $G_l[x,q] \geq 0$ for all $q \in S_\rho$. Considering Equation 24, $\theta_l[x,q]$ is super-regular if

$$f_l[x,q] = \frac{V[-x(1-q_l)]}{V[xq_l]} \leq \frac{\sum_{j\neq l} q_j d_j^*}{(1-q_l) d_l^*} \tag{25}$$

The right side of the above inequality consists of the nonnegative terms, so we have

$$\left(\sum_{j\neq l} q_j\right)\min_{j\neq l}\left(\frac{d_j^*}{d_l^*}\right) \leq \frac{1}{(1-q_l)}\sum_{j\neq l} q_j \frac{d_j^*}{d_l^*} \leq \left(\sum_{j\neq l} q_j\right)\max_{j\neq l}\left(\frac{d_j^*}{d_l^*}\right)$$

The above inequality can be rewritten as follows by substituting $\sum_{j\neq l} q_j$ by $(1-q_l)$.

$$\min_{j\neq l}\left(\frac{d_j^*}{d_l^*}\right) \leq \frac{\sum_{j\neq l} q_j \frac{d_j^*}{d_l^*}}{\sum_{j\neq l} q_j} \leq \max_{j\neq l}\left(\frac{d_j^*}{d_l^*}\right)$$

From Equation 24,it follows that $\theta_l[x,q]$ is super-regular if

$$f_l[x,q] \geq \max_{j\neq l}\left(\frac{d_j^*}{d_l^*}\right) \tag{26}$$

For more simplification, let employ logarithms. We define $\Delta(q,x) = \ln f_l[x,q]$. It has been shown in [55] that

$$-\int_0^x H'(u)du \leq \Delta(q,x) \leq -\int_{-x}^0 H'(u)du$$

$$H(u) = \frac{dH(u)}{du}, \quad H(u) = \ln V(u)$$

Therefore, we have

$$\frac{1}{V[x]} \leq \frac{V[-x(1-q_l)]}{V[xq_l]} \leq V[-x]$$

and

$$\frac{1}{V[x]} = \max_{j\neq l}\left(\frac{d_j^*}{d_l^*}\right) \tag{27}$$

Let $x^*$ be the value of $x$ for which the above equation is true. It is shown that if $d_j/d_l$ is smaller than 1 for all $j \neq l$, then there exists a value of $x > 0$ under which this equation is satisfied. By choosing $x = x^*$, Eq. 27 holds true, and thus $G_l[x,q] \geq 0$ for all $q \in S_\rho$ and $\theta_l[x,q]$ is super-regular. Therefore, Equation 22 is a sub-regular function satisfying the boundary conditions in Eq. 21. From Theorem 2, we can conclude that $\phi_l[x,q] \leq \Gamma_l(q) \leq 1$. From the definition of $\phi_l[x,q]$, we can see that given any $\varepsilon > 0$ there exists a positive constant $a^* < 1$ such that $1-\varepsilon \leq \phi_l[x,q] \leq \Gamma_l(q) \leq 1$ for all $0 < a \leq a^*$. As a result, the probability with which the algorithm DyTrust finds the trust path with the highest expected reliability is equal to 1 as $t \to \infty$, and so Theorem 1 is proved. □

**Theorem 2.** Let $(1-\varepsilon)$ be the probability with which the algorithm DyTrust converges to the trust path $\pi_l$. If $\underline{q}(t)$ is updated according to DyTrust, then for every $\varepsilon \in (0,1)$, there exists a learning parameter $a \in (\varepsilon, q)$ such that

$$\frac{xa}{e^{xa}-1} = \max_{j\neq l}\left(\frac{d_j}{d_l}\right) \tag{28}$$

where $1 - e^{-xq_l} = (1 - e^{-x})(1 - \varepsilon)$ and $q_l = [q_l(t)|t = 0]$.

*Proof.* It has been proved in [48] that if $d_j/d_l < 1$ for all $j \neq l$,then there exists a $x > 0$ under which Equation 27 is satisfied. Therefore, it is concluded that

$$\phi_l[x,q] \leq \Gamma_l(q) \leq \frac{1 - e^{-xq_l}}{1 - e^{-x}}$$

where $q_l$ denotes the initial choice probability of the optimal trust path $\pi_l$. From Theorem 1, it follows that for each $0 < a < a^*$ the probability of converging DyTrust to the trust path with the highest expected reliability is $(1 - \varepsilon)$ where $a^*(\varepsilon) \in (0,1)$. Therefore, we can conclude that

$$\frac{1 - e^{-xq_l}}{1 - e^{-x}} = 1 - \varepsilon \tag{29}$$

It is shown that for every $\varepsilon \in (0,1)$, there exists a value of $x$ under which Eq. 27 is satisfied, and so we have

$$\frac{x^* a}{e^{x^* a} - 1} = \max_{j \neq l} \left( \frac{d_j^*}{d_l^*} \right)$$

It is concluded that for every $\varepsilon \in (0,1)$, there exists a learning parameter $a \in (\varepsilon, q)$ under which the convergence probability of DyTrust to the most reliable trust path is greater than $(1 - \varepsilon)$. This completes the proof of the theorem. □

# 6. EXPERIMENS

In order to evaluate the performance of our proposed algorithm DyTrust, we conduct several expensive experiments on the real trust network dataset of Kaitiaki and compare the effectiveness of DyTrust with that of the well-known trust propagation algorithms, TidalTrust [11] and MoleTrust [41]. We also consider two algorithms proposed in our previous work on trust propagation [9], namely Min-MCFAvg and DLATrust, for the performance comparison. In the experiments, the path probability threshold $\mathcal{P}$ and the decay factor $\lambda$ are set to 0.9, and the threshold $K$ for the number of traversed paths is set to 10000. Experimental results are reported as an average of 10 independent runs. We use a system with the hardware configuration of Intel® Core i5 2.53 GHz and 4 GB RAM for all the experiments.

## 6.1 Experimental Design

The evaluation is done using a standard evaluation technique in machine learning: leave one out [56]. The trust link between source $v_s$ and target user $v_d$ is masked. Next, the trust weight $w_{sd}^{'}$ from $v_s$ to $v_d$ is calculated through trust propagation algorithms using the remaining trust links. While existing algorithms only use trust weights of the initial snapshot of trust network in the inference process, our proposed algorithm DyTrust considers the changes of trust weights during its running time. At last, a sample is taken from the trust weight of $v_s$ on $v_d$ as the actual trust weight with which the weight $w_{sd}^{'}$ calculated by the algorithms is compared. In this way, the actual trust weight is different from the trust weight masked from the initial snapshot. It is ensured that for any pair $v_s$ and $v_d$ of nodes the same actual trust weight is used in the evaluation process of all the algorithms.

We use two metrics of *coverage* and *prediction accuracy*, which have been widely used in the literature of trust [4,8,9,57,58]. Coverage represents the ability of an algorithm to provide a trust prediction and is computed as the percentage of trust links that are predictable for the algorithm. Prediction accuracy refers to the ability to predict that a user will be trusted or not. We use the following metrics for measuring the trust prediction accuracy:

*Mean absolute error (MAE).* This metric measures how close the predictions are to the actual trust weights and is computed as the average of the difference between the actual and calculated trust weights on the set of edges whose trust weight can be predicted (denoted as $E'$).

$$MAE = \frac{\sum_{e_{ij} \in E'} |w'_{ij} - w_{ij}|}{|E'|} \tag{30}$$

*Precision.* This metric measures the fraction of users who are predicted by the algorithm to be trusted and are actually trusted ones. Let $T_A$ be the number of trust links on which $v_s$ directly trusts $v_t$, and $T_B$ be the number of trust links on which $v_s$ trusts $v_t$ by the trust weight calculated through the algorithm. The Precision metric is computed as

$$Precision = \frac{T_A \cap T_B}{T_B} \tag{31}$$

*Recall.* This metric measures the fraction of users who are actually trusted and are successfully predicted by the algorithm. Recall is defined as

$$Recall = \frac{T_A \cap T_B}{T_A} \tag{32}$$

*Fscore.* Since there is a trade-off between precision and recall, Fscore is used to measure the accuracy using these two metrics jointly. The Fscore metric is computed as

$$Fscore = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{33}$$

### 6.1.1 Dataset

We test our proposed algorithm DyTrust on a real trust network dataset of Kaitiaki (www.kaitiaki.org.nz). The Kaitiaki dataset is a small trust network which contains 178 trust statements issued by 64 users on September 1, 2008 at four levels: Kaitiro, Te Hunga Manuhiri, Te Hunga Käinga, Te Komiti Whakahaere. Since we need trust weights to be real-valued, we initially assign 0.2 to Kaitiro, 0.6 to Te Hunga Manuhiri, 0.8 to Te Hunga Käinga, and 1 to Te Komiti Whakahaere. In this way, the initial snapshot of the Kaitiaki network is constructed. We then use the technique proposed in [59] to update trust weights over time. This technique has been widely utilized in the literature [8,9,58,60] to augment trust networks with trust weights which are continuous in [0,1]. In this technique, each user $v_i$ is assigned a quality measurement $q_i \in [0,1]$ which determines the probability that a statement provided by $v_i$ is true. Here, we consider a user's reputation as the quality of the user, namely for a user $v_i$ the quality $q_i$ is computed as the average of trust statements incoming in $v_i$. Since the trust weights are between 0 and 1, it is ensured that users' quality is also in [0,1]. After that, for any pair of users $v_i$ and $v_j$, the new trust weight $w_{ij}$ from $v_i$ to $v_j$ is uniformly chosen from $\left[\max\left(q_j - \delta_{ij},0\right),\min(q_j + \delta_{ij},1)\right]$, where $\delta_{ij} = \frac{1-q_i}{2}$ denotes a noise parameter which determines how accurate the user $v_i$ is at estimating the quality of the user $v_j$ that he is trusting.

## 6.2 Experimental Results

### 6.2.1 The impact of learning parameter

This experiment is conduct to study the impact of the learning parameter $a$ on the performance of our proposed algorithm DyTrust. For this purpose, we test DyTrust for different values of $a$ varying from 0.009 to 0.09 on the Kaitiaki dataset and for each value, we report in Table 1 the prediction accuracy of DyTrust in terms of MAE, Precision, Recall and FScore metrics. From the results of this table, we can see that the algorithm DyTrust obtains the best accuracy (with the highest FScore, and MAE close to the least MAE obtained for $a = 0.01$) for $a = 0.03$. Therefore, for the experiments that follow we set the value of the learning parameter $a$ to 0.03 in our proposed algorithm.

**Table 1.** The impact of learning parameter $a$ on the performance of DyTrust

| $a$ | MAE | Precision | Recall | FScore |
|---|---|---|---|---|
| 0.009 | 0.0847 | 0.9574 | 0.9887 | 0.9727 |
| 0.01 | **0.0830** | 0.9562 | 0.9898 | 0.9727 |
| 0.03 | 0.0834 | 0.9543 | **0.9943** | **0.9739** |
| 0.05 | 0.0836 | **0.9584** | 0.9887 | 0.9733 |
| 0.07 | 0.0849 | 0.9555 | 0.9921 | 0.9734 |
| 0.09 | 0.0845 | 0.9553 | 0.9887 | 0.9717 |

### 6.2.2 Evaluating different strategies

In this experiment, we validate the effectiveness of the proposed algorithm DyTrust in comparison with the other trust propagation algorithms in terms of the prediction accuracy and the running time. The algorithms used for the comparison are: TidalTrust, MoleTrust, Min-MCFAvg and DLATrust. Similar to DLATrust, our proposed algorithm DyTrust considers no limit for the length of trust paths. In contrast, the other three strategies are only based on shortest trust paths from source to target. We report the accuracy and running time of these algorithms on the Kaitiaki dataset in Table 2.

**Table 2.** Comparison between different strategies

| Strategy | MAE | Precision | Recall | FScore | Time (s) |
|---|---|---|---|---|---|
| TidalTrust | 0.1193 | 0.9705 | 0.9447 | 0.9574 | **0.0043** |
| MoleTrust | 0.1148 | 0.9705 | 0.9447 | 0.9574 | 0.0061 |
| Min-MCFAvg | 0.1355 | **0.9711** | 0.9668 | 0.9689 | 0.0050 |
| DLATrust | 0.1417 | 0.9710 | 0.9646 | 0.9678 | 5.1037 |
| DyTrust | **0.0834** | 0.9543 | **0.9943** | **0.9739** | 0.5477 |

From this table, we can see that our algorithm Dytrust yields the highest prediction accuracy compared to the others. While DyTrust considers the changes of trust weights during its trust propagation process, the other algorithms estimate the trustworthiness of a given target user only based on the initial snapshot of Kaitiaki. As a result, it is possible that, for example, the target user which has been estimated to be trusted by these algorithms is no longer trustworthy with respect to the trust variations occurred during their running time. Based on this, there is a trade-off between the computational complexity and the inference accuracy. It is worthy to mention that the additional computations required for considering the trust dynamicity are not too time consuming and affect the running time by some constant factor. As reported in Table 2, Dytrust has much less running time than DLATrust, while both algorithms consider all paths in the trust inference process. However, it takes more time in comparison with the algorithms using only shortest trust paths.

### 6.2.3 Testing the influence factors

*Trust Threshold.* We investigate the impact of the trust threshold on the prediction accuracy and coverage of the proposed algorithm DyTrust in comparison with the other trust propagation algorithms. For this purpose, we change the trust threshold in the range of [0.1,0.9] and for each threshold, we only use the trust statements of those direct neighbors that are reliable at or above the threshold in the trust inference process. The accuracy and coverage of our algorithm DyTrust for different trust thresholds are compared with those of the other algorithms in Figures 4(a-b) and 6(a).

According to the results of these figures, our proposed algorithm DyTrust provides the highest accuracy and coverage for all trust thresholds in comparison with the other algorithms. During the increase of the trust threshold, the MAE and FScore values remain about constant for all the algorithms, except for the threshold equal to 0.9. Increasing the trust threshold decreases the coverage of the algorithms, especially when the threshold is equal to 0.9. It indicates that the trust threshold should not be set to a large value, since for a larger threshold, fewer paths will be trusted.

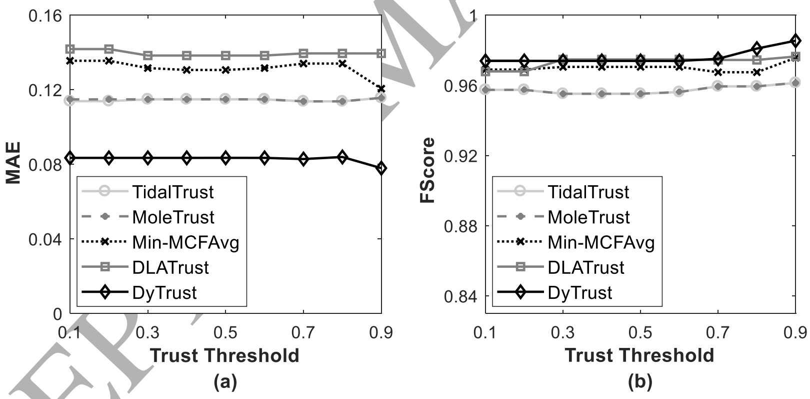


**Fig. 4.** The impact of trust threshold on the inference accuracy

*Maximum Length.* We examine the impact of the maximum path length on the performance of the proposed algorithm DyTrust compared to the other algorithms. The maximum length varies from 2 to 7. Figures 5(a-c) and 6(b) show the comparison of, respectively, MAE, FScore, running time and coverage of DyTrust with those of the other algorithms for different maximum lengths. Since the algorithms with which DyTrust is compared provide the same coverage for any length, in Figure 6(b) we report the coverage value of these algorithms only once labeled as “Others”.

From Figures 5(a) and (b), we can observe that for all the algorithms the MAE error at first increases along with the maximum length and then remains about constant. FScore of DyTrust increases when the maximum length varies from 2 to 3, while for the maximum length larger than 3, it goes down a little. We analyzed the reason and found that searching with a larger maximum length results in finding more paths which are of course longer and hence less reliable. This will decrease the accuracy of trust inference. In contrast, for the other algorithms under comparison the FScore value decreases with the increase of the maximum length until 4 and after that, it increases a little. Generally, DyTrust obtains the highest prediction accuracy for all the maximum lengths as compared to the other algorithms.

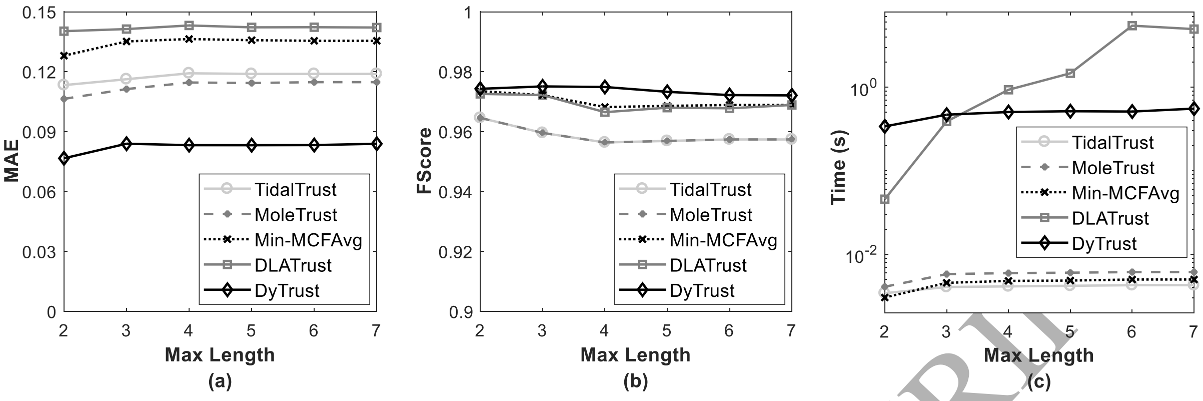


**Fig. 5.** The impact of maximum path length on the inference accuracy and time

As shown in Figure 5(c), increasing the maximum length obviously increases the running time for all the algorithms, especially when it changes from 2 to 3. The increase is much faster for the DLATrust algorithm. According to the results of Figure 6(b), the coverage is also increased with increasing the maximum length. The reason is that more new trust paths can be found as the maximum length gets larger. Moreover, for all the maximum lengths our algorithm DyTrust provides the same coverage as the other algorithms. It indicates that DyTrust always successfully discovers reliable trust paths satisfying the path length constraint between any pair of nodes.

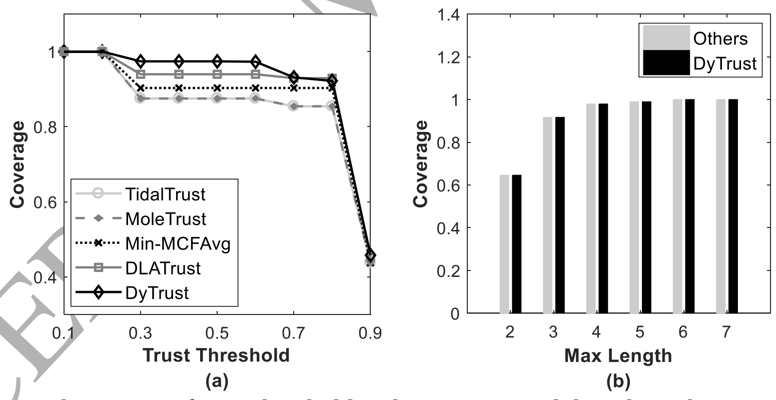


**Fig. 6.** The impact of trust threshold and maximum path length on the coverage

## 7. DISCUSSION AND CONCLUSION

In this paper, we proposed a dynamic trust propagation algorithm based on distributed learning automata, called DyTrust, for inferring trust in stochastic trust networks. Since trust changes over time as a result of repeated direct interactions between users, trust networks can be modelled as stochastic graphs with continuous time-varying edge weights. Even though the dynamic nature of trust has been universally accepted in literature, existing trust propagation algorithms do not take the dynamicity of trust into consideration. These algorithms take an instant snapshot of trust network and then deterministically infer trust in the network snapshot. Due to being time consuming of trust propagation algorithms, it is highly probable that trust weights change during the algorithms' running time and therefore the estimated trust values will not have enough accuracy. Our proposed algorithm DyTrust is the first to address the dynamicity property of trust in the domain of trust inference. Considering the changes of trust weights in time, DyTrust finds the most reliable trust path to each neighboring user of target and estimates the trust value of the target based on its neighbors' reliability.

In order to validate the effectiveness of our algorithm DyTrust, we conducted extensive experiments with the real trust network dataset, Kaitiaki and compared the results of DyTrust with those of the well-known trust propagation algorithms, as well as with those of our previous algorithms on trust propagation [9]. We found that by considering the temporal variation of trust, DyTrust outperforms the existing algorithms in terms of the prediction accuracy. Since DyTrust uses all trust paths in the inference process, it needs more time in comparison with the algorithms using only shortest paths. However, the running time of DyTrust is much less than that of DLATrust which is also based on all trust paths.

We also investigated the impact of the influence factors including trust threshold and maximum path length on the performance of our algorithm DyTrust as compared to the other algorithms. From the results, we found that while increasing trust threshold decreases the prediction coverage, the accuracy of the algorithms remains about constant. Moreover, we observed that DyTrust always has the highest accuracy and coverage for all trust thresholds comparing to the others. Finally, by testing different maximum lengths, we found that increasing the maximum length will decrease the accuracy of DyTrust, although it always obtains the highest accuracy for all lengths in comparison with the other algorithms. We also observed that the increase of the maximum length increases the running time and prediction coverage for all the algorithms under comparison.

In future work, we plan to carry out more extensive experiments and use larger trust network datasets for evaluating our dynamic trust propagation algorithm. In addition, we will attempt to extend our system model by defining the time scales and the model of trust evolution over time.

## REFERENCES

[1] Y.A. Kim, An enhanced trust propagation approach with expertise and homophily-based trust networks, Knowledge-Based Syst. 82 (2015) 20–28. doi:10.1016/j.knosys.2015.02.023.

[2] X. Liu, Y. Wang, S. Zhu, H. Lin, Combating Web spam through trust-distrust propagation with confidence, Pattern Recognit. Lett. 34 (2013) 1462–1469. doi:10.1016/j.patrec.2013.05.017.

[3] S. Lyu, J. Liu, M. Tang, Y. Xu, J. Chen, Efficiently Predicting Trustworthiness of Mobile Services Based on Trust Propagation in Social Networks, Mob. Networks Appl. 20 (2015) 840–852. doi:10.1007/s11036-015-0619-y.

[4] W. Jiang, G. Wang, J. Wu, Generating trusted graphs for trust evaluation in online social networks, Futur. Gener. Comput. Syst. 31 (2014) 48–58.

[5] G. Wang, J. Wu, Multi-dimensional evidence-based trust management with multi-trusted paths, Futur. Gener. Comput. Syst. 27 (2011) 529–538. doi:10.1016/j.future.2010.04.015.

[6] F.J. Ortega, J.A. Troyano, F.L. Cruz, C.G. Vallejo, F. Enríquez, Propagation of trust and distrust for the detection of trolls in a social network, Comput. Networks. 56 (2012) 2884–2895. doi:10.1016/j.comnet.2012.05.002.

[7] Y.A. Kim, H.S. Song, Strategies for predicting local trust based on trust propagation in social networks, Knowledge-Based Syst. 24 (2011) 1360–1371. doi:10.1016/j.knosys.2011.06.009.

[8] 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.

[9] M. Ghavipour, M.R. Meybodi, Trust propagation algorithm based on learning automata for inferring local trust in online social networks, Knowledge-Based Syst. (2017). doi:10.1016/j.knosys.2017.06.034.

[10] S. Al-Oufi, H.-N. Kim, A. El Saddik, A group trust metric for identifying people of trust in online social networks, Expert Syst. Appl. 39 (2012) 13173–13181.

[11] J.A.J.A. Golbeck, Computing and applying trust in web-based social networks, (2005). doi:10.1017/CBO9781107415324.004.

[12] D.M. Rousseau, S.B. Sitkin, R.S. Burt, C. Camerer, Not so different after all: A cross-discipline view of trust, Acad. Manag. Rev. 23 (1998) 393–404.

[13] W. Sherchan, S. Nepal, C. Paris, A survey of trust in social networks, ACM Comput. Surv. 45 (2013) 47.

[14] S.D. Kamvar, M.T. Schlosser, H. Garcia-Molina, The eigentrust algorithm for reputation management in p2p networks, in: Proc. 12th Int. Conf. World Wide Web, ACM, 2003: pp. 640–651.

[15] G. von Laszewski, B.K. Alunkal, I. Veljkovic, Toward reputable grids, Scalable Comput. Pract. Exp. 6 (2001).

[16] R. Wishart, R. Robinson, J. Indulska, A. Jøsang, SuperstringRep: reputation-enhanced service discovery, in: Proc. Twenty-Eighth Australas. Conf. Comput. Sci. 38, Australian Computer Society, Inc., 2005: pp. 49–57.

[17] C.J. Fung, J. Zhang, I. Aib, R. Boutaba, Dirichlet-based trust management for effective collaborative intrusion detection networks, IEEE Trans. Netw. Serv. Manag. 8 (2011) 79–91.

[18] S. Song, K. Hwang, R. Zhou, Y.-K. Kwok, Trusted P2P transactions with fuzzy reputation aggregation, IEEE Internet Comput. 9 (2005) 24–34.

[19] Y. Zhang, Y. Fang, A fine-grained reputation system for reliable service selection in peer-to-peer networks, IEEE Trans. Parallel Distrib. Syst. 18 (2007).

[20] R. Jurca, B. Faltings, An incentive compatible reputation mechanism, in: E-Commerce, 2003. CEC 2003. IEEE Int. Conf., IEEE, 2003: pp. 285–292.

[21] B. Yu, M.P. Singh, Distributed reputation management for electronic commerce, Comput. Intell. 18 (2002) 535–549.

[22] R. Chen, F. Bao, M. Chang, J.-H. Cho, Dynamic trust management for delay tolerant networks and its application to secure routing, IEEE Trans. Parallel Distrib. Syst. 25 (2014) 1200–1210.

[23] L. Xiong, L. Liu, A reputation-based trust model for peer-to-peer e-commerce communities, in: E-Commerce, 2003. CEC 2003. IEEE Int. Conf., IEEE, 2003: pp. 275–284.

[24] L. Xiong, L. Liu, Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities, IEEE Trans. Knowl. Data Eng. 16 (2004) 843–857.

[25] J. Zhang, R. Cohen, Evaluating the trustworthiness of advice about seller agents in e-marketplaces: A personalized approach, Electron. Commer. Res. Appl. 7 (2008) 330–340.

[26] Z. Su, M. Li, X. Fan, X. Jin, Z. Wang, Research on trust propagation models in reputation management systems, Math. Probl. Eng. 2014 (2014). doi:10.1155/2014/536717.

[27] R. Zhou, K. Hwang, Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing, IEEE Trans. Parallel Distrib. Syst. 18 (2007).

[28] A. Abdul-Rahman, S. Hailes, Supporting trust in virtual communities, in: Syst. Sci. 2000. Proc. 33rd Annu. Hawaii Int. Conf., IEEE, 2000: p. 9–pp.

[29] H. Liu, E.-P. Lim, H.W. Lauw, M.-T. Le, A. Sun, J. Srivastava, Y. Kim, Predicting trusts among users of online communities: an epinions case study, in: Proc. 9th ACM Conf. Electron. Commer., ACM, 2008: pp. 310–319.

[30] J. Caverlee, L. Liu, S. Webb, Towards robust trust establishment in web-based social networks with socialtrust, in: Proc. 17th Int. Conf. World Wide Web, ACM, 2008: pp. 1163–1164.

[31] Z.-P. Fan, W.-L. Suo, B. Feng, Y. Liu, Trust estimation in a virtual team: A decision support method, Expert Syst. Appl. 38 (2011) 10240–10251.

[32] J. Witkowski, Trust mechanisms for online systems, in: IJCAI Proceedings-International Jt. Conf. Artif. Intell., 2011: p. 2866.

[33] G. Vogiatzis, I. MacGillivray, M. Chli, A probabilistic model for trust and reputation, in: Proc. 9th Int. Conf. Auton. Agents Multiagent Syst. Vol. 1-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, 2010: pp. 225–232.

[34] A. Josang, R. Ismail, The beta reputation system, in: Proc. 15th Bled Electron. Commer. Conf., 2002: pp. 2502–2511.

[35] A. Josang, J. Haller, Dirichlet reputation systems, in: Availability, Reliab. Secur. 2007. ARES 2007. Second Int. Conf., IEEE, 2007: pp. 112–119.

[36] A.J.I. Jones, J. Pitt, On the classification of emotions and its relevance to the understanding of trust, in: Proc. Work. Trust Agent Soc. 10th Int. Conf. Auton. Agents Multi-Agent Syst. (AAMAS 2011), 2011: pp. 69–82.

[37] L.-L. Jiang, M. Perc, Spreading of cooperative behaviour across interdependent groups, Sci. Rep. 3 (2013).

[38] Q. Li, M. Chen, M. Perc, A. Iqbal, D. Abbott, Effects of adaptive degrees of trust on coevolution of quantum strategies on scale-free networks, Sci. Rep. 3 (2013).

[39] Y. Ren, M. Li, Y. Xiang, Y. Cui, K. Sakurai, Evolution of cooperation in reputation system by group-based scheme, J. Supercomput. (2013) 1–20.

[40] M. Perc, P. Grigolini, Collective behavior and evolutionary games–An introduction, Chaos, Solitons & Fractals. 56 (2013) 1–5.

[41] P. Avesani, P. Massa, R. Tiella, A trust-enhanced recommender system application: Moleskiing, Sac. (2005) 1589–1593.

[42] 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.

[43] 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.

[44] Z. Su, L. Liu, M. Li, X. Fan, Y. Zhou, ServiceTrust: Trust management in service provision networks, in: Serv. Comput. (SCC), 2013 IEEE Int. Conf., IEEE, 2013: pp. 272–279.

[45] V.G. Vydiswaran, C. Zhai, D. Roth, Content-driven trust propagation framework, in: Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., ACM, 2011: pp. 974–982.

[46] M.-H. Jang, C. Faloutsos, S.-W. Kim, U. Kang, J. Ha, Pin-trust: Fast trust propagation exploiting positive, implicit, and negative information, in: Proc. 25th ACM Int. Conf. Inf. Knowl. Manag., ACM, 2016: pp. 629–638.

[47] M.A.L. Thathachar, P.S. Sastry, Networks of learning automata: Techniques for online stochastic optimization, Springer Science & Business Media, 2011.

[48] K.S. Narendra, M.A.L. Thathachar, Learning automata: an introduction, Courier Corporation, 2012. doi:10.1109/TSMCB.2002.1049606.

[49] M. Ghavipour, M.R. Meybodi, A streaming sampling algorithm for social activity networks using fixed structure learning automata, Appl. Intell. (2017). doi:10.1007/s10489-017-1005-1.

[50] M. Ghavipour, M.R. Meybodi, Irregular cellular learning automata-based algorithm for sampling social networks, Eng. Appl. Artif. Intell. 59 (2017) 244–259. doi:10.1016/j.engappai.2017.01.004.

[51] M. Ghavipour, 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.

[52] J.A. Torkestani, M.R. Meybodi, Mobility-based multicast routing algorithm for wireless mobile Ad-hoc networks: A learning automata approach, Comput. Commun. 33 (2010) 721–735.

[53] H. Beigy, M.R. Meybodi, Utilizing distributed learning automata to solve stochastic shortest path problems, Int. J. Uncertainty, Fuzziness Knowledge-Based Syst. 14 (2006) 591–615.

[54] M.A.L. Thathachar, K.R. Ramakrishnan, A hierarchical system of learning automata, IEEE Trans. Syst. Man, Cybern. 11 (1981) 236–241.

[55] S. Lakshmivarahan, M.A.L. Thathachar, Bounds on the convergence probabilities of learning automata, IEEE Trans. Syst. Man, Cybern. A Syst. Humans. 6 (1976) 756–763.

[56] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: Ijcai, 1995: pp. 1137–1145.

[57] P. Massa, P. Avesani, Trust metrics on controversial users: Balancing between tyranny of the

majority, Int. J. Semant. Web Inf. Syst. 3 (2007) 39–64.

[58] S. Shekarpour, S.D. Katebi, Modeling and evaluation of trust with an extension in semantic web, Web Semant. Sci. Serv. Agents World Wide Web. 8 (2010) 26–36.

[59] M. Richardson, R. Agrawal, P. Domingos, Trust management for the semantic web, in: Int. Semant. Web Conf., Springer, 2003: pp. 351–368.

[60] W. Jiang, J. Wu, G. Wang, On selecting recommenders for trust evaluation in online social networks, ACM Trans. Internet Technol. 15 (2015) 14.