Applied Soft Computing 118 (2022) 108475

Contents lists available at ScienceDirect

Applied Soft Computing

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

An automata algorithm for generating trusted graphs in online social networks

Nina Fatehi [a], Hadi Shahriar Shahhoseini [a], Jesse Wei [b], Ching-Ter Chang [c,d,e,*]

[a] School of Electrical Engineering, Iran University of Science and Technology, Tehran, Iran
[b] School of Arts and Sciences, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, USA
[c] Department of Information Management, Chang Gung University, 259 Wen-Hwa 1st Road, Kwei-Shan, Tao-Yuan 333, Taiwan, ROC
[d] Clinical Trial Center, Chang Gung Memorial Hospital at Linkou, Taoyuan, Taiwan, ROC
[e] Department of Industrial Engineering and Management, Ming Chi University of Technology, Taiwan, ROC

ARTICLE INFO

Article history:
Received 29 January 2021
Received in revised form 19 October 2021
Accepted 12 January 2022
Available online 24 January 2022

Keywords:
Online social network
Trust
Evaluation
Learning automata

ABSTRACT

Online social networks (OSNs) are becoming a popular tool for people to socialize and keep in touch with their friends. OSNs need trust evaluation models and algorithms to improve users' quality of service and quality of experience. Graph-based approaches make up a major portion of existing methods, in which the trust value can be calculated through a trusted graph. However, this approach usually lacks the ability to find all trusted paths, and needs to put some restrictions to admit the process of finding trusted paths, causing trusted relations to be unreachable and leading to reduced coverage and accuracy. In this paper, graph-based and artificial intelligence approaches are combined to formulate a hybrid model for improving the coverage and accuracy of OSNs. In this approach, a distributed learning automata, which can be used to find all trusted relations without limitation, is employed instead of well-known graphic-based searching algorithms such as breadth-first search. Simulation results, conducted on real dataset of Epinions.com, illustrate an improvement of accuracy and coverage in comparison with state-of-the-art algorithms. The accuracy of the proposed algorithm is 0.9398, a 6% increase in accuracy over existing comparable algorithms. Furthermore, by the successful removal of imposed restrictions in the existing searching process for finding trusted paths, this algorithm also leads to a 10% improvement in coverage, reaching approximately 95% of all existing trusted paths.

1. Introduction

Online social networks (OSNs) are becoming major platforms for exploring social relations. In OSNs, hundreds of millions of users share their daily activities with others in various forms such as text, audio, photos, or videos. In OSNs, to design and conduct interactive and effective activities for business purposes, organizers need to assess whether a particular user or a product can be trusted or not. An OSN is represented by a graph, in which nodes are users and edges describe relations between users. Users in OSNs can be either individuals or organizations, while relations demonstrate friendships, ideas and even trades. An OSN is a virtualized form of the physical social structure in the real world. An E-mail system can be a good example of an initial form of social networks. However, with the evolution of social networks and the Internet, more complex online platforms like Facebook and Twitter were introduced and developed. In the past, users were mostly content consumers, yet now they are the main content generation source in OSNs.

An OSN includes three main entities: users, connections between users and information that flows in the network [1]. Users are deemed as the basis for the other entities. That is, users build the connections by interacting with each other, and also generate and share their own content, resulting in an information flow throughout the network. The OSNs are constantly developing as more and more users are joining them; consequently, extensive connections are built and large quantities of user-generated contents are diffused in the OSNs. In such large-scaled networks with large quantities of information, an important issue is how to evaluate and measure the trust levels between entities. In other words, the main objective of most existing investigations in the trust literature is to find the appropriate answer for Alice's question: "Can I trust Bob on this particular topic or not?" [1] (see Fig. 1).

* Corresponding author at: Department of Information Management, Chang Gung University, 259 Wen-Hwa 1st Road, Kwei-Shan, Tao-Yuan 333, Taiwan, ROC.
E-mail addresses: nina.fatehi@wayne.edu (N. Fatehi), shahhoseini@iust.ac.ir (H.S. Shahhoseini), jessew13@email.unc.edu (J. Wei), chingter@mail.cgu.edu.tw (C.-T. Chang).

https://doi.org/10.1016/j.asoc.2022.108475
1568-4946/

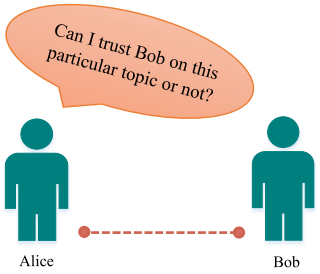


Fig. 1. The main objective in trust evaluation.

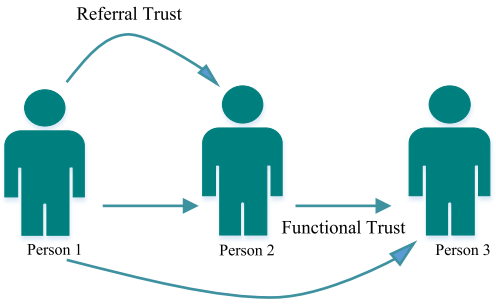


Fig. 2. Propagable property of trust.

The relationships and interactions among users in OSNs are more complicated due to the lack of real communications in OSNs, namely by virtual conversations. Without identified trust links in a network, daily social life would be easily misguided. Recognizing trust relationships in OSNs is a difficult computational problem and it theoretically requires algorithms of exponential growth in the calculation, not only because of the complex structure of the OSN but also because of the trust complexity itself. In addition, due to the sensitive nature of such data, it is essential to protect it from inauthentic entities that can threaten users' privacy in various forms. In summary, the study of trust relationship in OSNs is critical for scheduling online activities and delivering online services, and yet this is also technically difficult, particularly for the calculation of trust paths. As such, trust evaluation as a tool of decision making is singled out, attracting more and more attention in recent years [2], and it is widely utilized in various fields such as recommender systems and decision-making problems [3–5], peer-to-peer systems [6], internet of things [7], and OSN [8,9].

Accuracy and coverage are two prominent metrics in trust evaluation. The former represents the algorithm's ability to predict whether a user is trustworthy or not. Although computing the highest trust value is important, it is not the ultimate goal in trust evaluation. The decisive aim is to have higher confidence in the reliability of the estimated trust value. In other words, increasing trust accuracy ensures the reliability of computed trust value. Coverage illuminates algorithms' prediction ability and it is measured by the percentage of predictable trust relations within all that exist. Having a good coverage is important due to its role in increasing both the breadth and the depth of searching, avoiding any situation in which the algorithm fails to explore existing trusted paths between users. Therefore, increasing coverage requires an algorithm that can find the trusted links between the users in a network. Trust is a context-oriented concept and thus various definitions and categories have been introduced for it. Golbeck's definition will be adopted in this paper. According to Golbeck, trust is "a commitment to an action, based on a belief that the future actions of that person will lead to a good outcome". [10] Properties of trust have been categorized from different perspectives [11]. Two computational properties of trust are propagability and composability. The propagable property illustrates that trust relationships can be disseminated in a chain of relations through the transitive property, as shown in Fig. 2. Assume that person 1 trusts person 2 and person 2 trusts person 3. Though person 1 may not even know person 3, person 1 can place some trust in person 3 through person 2 by summing up the trust of person 1 in person 2 and the trust of person 2 in person 3. Consequently, person 1 can trust person 3 [12]. This property would make it possible to evaluate trust through the chain of relations. There is usually more than one path between two users, e.g., there may be several chains of relations between the trustor and the trustee. Therefore, trust values obtained from all paths are required to be composed by the trustee to evaluate the trustor. Namely, the composable property indicates the integration of trust values obtained from multiple paths into a unique value [13].

Recommendation is a closely related concept with trust, and it is defined as a method that allows opinions to be propagated within an OSN. Recommendation introduces two kinds of trust in a trusted graph: referral trust and functional trust. Referral trust is a trust that is computed out of the recommendations of nodes from the start node to a direct neighbor of the target node. Functional trust exists between the direct neighbors of the target node and the target node. The difference between referral trust and functional trust is exhibited in Fig. 2 [14].

Various models are proposed for trust evaluation in the literature. These models fall into four groups.

(1) Approaches based on Dempster–Shafer theory and subjective logic. This approach is proposed to cope with subjectivity of trust, and infer uncertainty [15].

(2) Mathematical tools-based approaches (probability statistics and fuzzy logic). In this approach, trust evaluation is commutated utilizing probability functions, such as Bayesian model [16]. However, the main challenge is how to decrease the computational complexity of reasoning rules in fuzzy logic-based models [11].

(3) Artificial intelligence and information theory-based approaches. In this group, trust evaluation models usually employ learning methods for estimating trust in an OSN [17]. The main issue is that the model itself is considered more important than the available information for trust evaluation. This increases the risk of subjectivity of trust in such models.

(4) Graph-based approaches [18–20]. Graph-based approaches make up a large portion of modern methods. In graph-based models, an OSN is represented by a graph, where a node denotes a user and an edge represents the connection between two users. Trust evaluation in graph-based approaches is done through a trusted graph. A trusted graph includes a trustor, a trustee, and at least one recommender and is constructed of numerous trusted paths between the trustor and the trustee. Moreover, a trusted path is formed from recommendations. In this approach, trusted paths between trustor and trustee need to be found. Most of the works based on the graph-based approaches utilize graph-simplification for evaluating trust [21]. Graph-simplification utilizes an amalgamation of trusted paths for trust evaluation. They simplify the trusted graph into node/link-disjoint or a simple directed serial/parallel graph [22]. Graph-based simplification models apply computational properties of trust on finding trusted paths. Nevertheless, the propagable property leads the model to confront a challenge of finding the paths with appropriate length. In other words, large scale OSNs make it impossible to find all paths between two users by the common search algorithms such as Breadth-First Search (BFS). Thus, finding smaller paths

satisfying certain imposed constraints on the process becomes a practically feasible approach for trust evaluation. By setting a higher limitation on path length, smaller and fewer paths are achieved. On the other hand, a larger length causes a decrease in trust accuracy. As a result, trust accuracy and coverage are two important metrics in trust evaluation process. In artificial intelligence, trust evaluation is taken as a problem of learning or a problem of decision support system [11,23].

The graph-based approach in trust evaluation literature is a practically executable algorithm by putting restrictions on the search process of finding trusted paths between users, which usually is accomplished using the BFS algorithm. However, the computational complexity of BFS could exponentially increase with a rise in the size of the OSN if the number of edges is increased. In addition, such limitations cause the trust accuracy and coverage to decrease. Learning algorithms are stochastic process that can explain the behavior of a subject in a feedback interaction [24]. In such a feedback interaction, the agent will learn and converge to the desirable behavior after many interactions with the environment and thus there is no need to restrict the search process in learning algorithms in contrast to the BFS, a deterministic algorithm, which requires to impose restrictions to avoid computation complexity. Fortunately, combining the graph-based approach with a learning algorithm can overcome the limitation of the graph-based approach. Besides, most of the proposed models are based on just one approach, and specifically, hybrid approaches are rarely seen in the studies, leaving a research gap in the literature. Therefore, a major contribution of this paper is to combine the graph-based approach and the artificial intelligence-based approach as a hybrid model to significantly overcome the challenge of path length in graph-based algorithms by finding all trusted paths between users that satisfy certain imposed conditions in the network and subsequently increase the coverage and trust accuracy in OSNs. In this model, firstly, the neighborhood is defined according to "social distance". In this way, each user's neighbors are determined. Then, in order to find trusted paths between each pair of users, learning automata is utilized in the trust network instead of BFS. In order to learn next reliable neighbors along a trusted path, each node of the trust network is assigned a learning automaton, and a trusted graph including users and their neighbors is generated for trust evaluation.

In the end, to determine the effectiveness of the proposed algorithm, experiments are carried out on a real dataset from Epinions.com, and the results are compared with the results derived from other established graph-based algorithms. Experiments are provided to show that the proposed algorithm performs better than existing graph-based models in terms of trust accuracy and coverage, with an over-performance of 6% and 10%, respectively (Section 5).

The rest of the paper is organized as follows: related works are briefly outlined in Section 2. Section 3 presents learning automata and distributed learning automata. In Section 4, the proposed model is discussed. Experimental evaluation, as well as results is discussed in Section 5. Finally, in Section 6 the main work is summarized and future work is outlined.

2. Related works

In this section, we first review some existing related works on trust evaluation. Then, we discuss the challenges of these models and present our approach in the study.

Golbeck [10] proposed a linear computational graph-based model named TidalTrust, which discovers trusted paths from a trustor to a trustee by applying a BFS to the trust network. The final estimated trust value from the trustor to the trustee is computed based on an average of the trust value of the strongest path among all the shortest paths as follows:

$$t_{sd} = \frac{\sum_{j \in Nl_s, t_{sj} \geq th} t_{sj} t_{jd}}{\sum_{j \in Nl_s, t_{sj} \geq th} t_{sj}} \quad (1)$$

where Nl_s is the neighbor list of the source node, and th represents the trust threshold. In TidalTrust, only the shortest paths with highest trust values are found due to the limitation of BFS, causing a decrease in trust accuracy and a blind removal of some trustworthy relations, resulting in a decline in the coverage.

MoleTrust was proposed by Massa et al. [13]. In the first step, it ranks users according to their nearest distance from the trustor; in this way, cycles are removed. The second step is trust computation. For this purpose, it first chooses the users who are located a step away from the trustor, then those that are two steps away, and so on. Then, the shortest path from the trustor to the trustee is established. This model aggregates trust values of all reliable direct neighbors of the trustee by calculating the weighted average as follows:

$$t_d = \frac{\sum_{j \in dNl_d} t_j t_{jd}}{\sum_{j \in dNl_d} t_j} \quad (2)$$

where dNl_d denotes the reliable direct neighbors of the target node and t_j is the trust value of node j. Considering only shortest paths gives rise to a decline in trust accuracy and coverage.

MeTrust is a multi-dimensional evidence-based trust evaluation system that was proposed by Wang and Wu [25]. This model computes trust in three levels. In the node layer, trust is considered multidimensional. Each dimension can be valued a different weight by each user. The path layer controls the trust rate, which decreases during combination. In graph layer, three algorithms are used (i.e., GraphReduce, GraphAdjust, and WeightedAverage) to reduce the complexity of the trusted graphs. Jiang et al. [1], proposed SWTrust as a framework for preprocessing OSNs and generating trusted graphs. SWTrust implements the advantages of "small-world network" and "strength of weak ties" theories. Trust values are calculated with the users' active domains information. This type of information is much more objective in comparison with the ratings. Neighbors of each user are categorized into three groups based on their social distance from the user: local neighbors, longer contacts, and farthest contacts. Then, trusted paths are established by applying width-adjustable BFS to the trust network. In this algorithm, a parameter k is defined to determine the number of hops and thus controls the width of the search. Kim [26] proposed a trust evaluation algorithm with a combination of homophily-based trust and expertise-based trust in order to increase the accuracy and coverage in sparse online social networks; however, it confines the width of search to overcome the problem of computation complexity while traversing all nodes and finding all paths between each pair of users.

In the graph-based approach, trust is computed through a trusted graph. In this approach, the computational property of the trust is applied to find trusted paths between two users. The path length is a challenge in this approach. Due to computational complexity, it is not possible to consider all paths with different lengths. However, small distances between two users result in fewer paths to be found. Subsequently, less trust evidence will be available. On the other hand, a larger length may decrease accuracy, as it has been directly stated by Josang et al. [27] that "trust can decrease during the propagation process". In other words, a longer referral trust chain contributes to a weaker trust prediction. In this regard, Golbeck [10] conducted several experiments and found that as the path length increases, precision decreases. Therefore, she concluded that shortest paths would predict trust with higher accuracy and only used the shortest paths with highest trust values for inferring trust in the proposed trust model.

Lesani and Montazeri [28] stated that information obtained from a longer trust chain with higher trust is more accurate than a shorter trust chain with lower trust. Consequently, in order to have a more reliable trust estimation, both short and long paths should be considered. Cho et al. [29] tried to find the optimal path length by compromising between the trust availability and the path reliability, but the path length is still inappropriate. Kim and Song [30] indicated that trust evaluation by considering all paths is more precise than the current approach based on only the shortest paths or considering specific path length. OSNs are networks containing an extensive number of users; finding all the possible paths between two users using deterministic searching algorithms, such as BFS is impractical because increasing the number of users increases the computational complexity as well. The BFS requires $O(|n| + |E|)$ time in computation [31], where n and E present the number of nodes and edges in the graph, respectively. Therefore, researchers have to impose constraints on the process of discovering trusted paths. For instance, Jiang et al. [1] limits the width of the BFS; in TidalTrust [10] only the shortest paths with highest trust are considered. In MoleTrust [13], the number of nodes from the source to the target is restricted. Some trusted relations may be unreachable due to this limitation; consequently, coverage decreases. Also, it may omit some paths with high reliability, which leads to a decline in accuracy.

With regards to the path length challenge and the size of the OSN, processing and analyzing this amount of generated data in OSN using statics measures and traditional mathematical tools is a sophisticated task. Implementing artificial intelligence methods to the typical methods of evaluating trust can improve the efficiency of the proposed algorithms. As a result, in this paper, a hybrid model is proposed by combining graph-based and artificial intelligence-based approaches using learning automata to discover the trusted paths between two users instead of using the BFS. The learning automata not only learns the paths to reach the target but also identifies those users who probably are unreliable and removes them. In other words, this approach does not need to check all paths and also does not omit any paths blindly. Hence, even though the size of the network is increasing, the proposed algorithm is still significant. Consequently, the accuracy and coverage of the OSNs do not decrease.

It is worth mentioning that in OSNs, selecting neighbors towards the target user is generally a random work that has a large amount of uncertainty. Learning automata is useful for these applications. Furthermore, it needs no prior information, except a small amount of feedback from the environment, and its simple structure makes implementation possible in both software and hardware. These advantageous properties of learning automata make it very useful for various applications [32].

3. Learning automata

Interests in Learning Automata (LA) began by Tsetlin and his colleagues in the early 1960s in the Soviet Union [33]. Interested in biological systems modeling, he introduced a deterministic automaton that is incorporated with a random environment as a learning model. A LA [24] is an abstract self-organized decision-making unit that improves its own performance by repeating interactions with the environment. The LA learns how to select the optimal action based on the received response from the environment. The environment and automaton are connected in a feedback arrangement as shown in Fig. 3.

The output of the environment $\beta(t)$ provides the input to the automaton, and the action of the automaton $\alpha(t)$ forms the input of the environment. An environment is defined by a triple $\{\alpha, c, \beta\}$, where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ indicates a finite input set

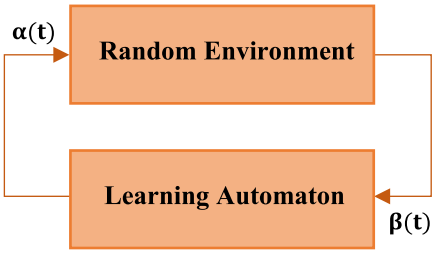


Fig. 3. The feedback connection between environment and learning automaton.

of the environment, $\beta = \{\beta_1, \beta_2\}$ is a binary output set and $c = \{c_1, c_2, \ldots, c_r\}$ represents the set of penalty probabilities, where c_i corresponds to the input action α_i. In a non-stationary environment, unlike the stationary environment, the penalty probabilities vary over time. Various models were provided based on the output of the environment: Models in which the output can take only two binary values (0 or 1) are referred to as P-models. In Q-model, the output can take more than only one or two values. In other words, it is allowed to take a finite number of the values in the interval [0, 1]. In S-models, environments allow a continuous number of values in the interval [0, 1]. The stochastic learning automata is categorized into two major families: variable structure learning automata (VSLA) and fixed structure learning automata (FSLA). A VSLA is represented by a quadruple $\{\alpha, c, \beta, T\}$ where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the actions set of learning automata, $\beta = \beta_1, \beta_2, \ldots, \beta_r$ indicates the automaton's set of inputs, $p = \{p_1, p_2, \ldots, p_r\}$ stands for the set of action probability vector where p_i is the probability of action α_i to be chosen, and T is the learning algorithm that updates the action probability vector $p(n+1) = T[\alpha(n), \beta(n), p(n)]$ based on the response of the environment. The inputs to T are the chosen action, response of the environment and probability vector of the action at instant n. The number of actions available in VSLA changes over time. In other words, it randomly chooses the action from the finite subset of available actions named $V(n)$ in each instant n. Let $K(n)$ equal to the sum of the available actions' probabilities in $V(n)$. Before selecting an action, the available action probability vector $p(n)$ is scaled as given in Eq. (4).

$$K(n) = \sum_{\alpha_i \in V(n)} p_i(n) \tag{3}$$

$$\hat{p}(n) = prob[\alpha(n) = \alpha_i | \alpha \in V(n)] = \frac{p_i(n)}{K(n)} \tag{4}$$

Then, one of the actions of the LA from $V(n)$ is randomly chosen, depending on the scaled action probability vector $\hat{p}(n)$. The environment receives the chosen action as its input and gives a reinforcement signal to the automaton. The vector $\hat{p}(n)$ based on the reinforcement signal is updated as given in Eq. (5) if the received response is favorable. Otherwise, the vector $\hat{p}(n)$ is updated as given in Eq. (6).

$$\begin{cases} \hat{p}_i(n+1) = \hat{p}_i(n) + a.(1 - \hat{p}_i(n)) \\ \hat{p}_j(n+1) = (1-a).\hat{p}_j(n) \quad \forall j\, j \neq i \end{cases} \tag{5}$$

$$\begin{cases} \hat{p}_i(n+1) = (1-b).\hat{p}_i(n) \\ \hat{p}_j(n+1) = \dfrac{b}{r-1} + (1-b).\hat{p}_j(n) \quad \forall j\, j \neq i \end{cases} \tag{6}$$

In the end, the LA rescales the action probability vector $\hat{p}(n)$ according to Eq. (7).

$$p_i(n+1) = \hat{p}_i(n+1).K(n) \tag{7}$$

In Eq. (4), r represents the number of actions, and a and b indicate reward and penalty parameter, respectively, which

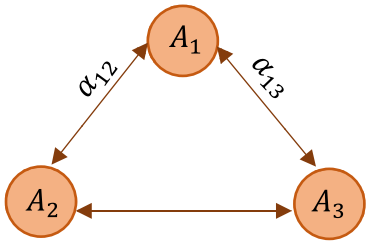


Fig. 4. A distributed learning automata.

Table 1
Notations explanation.

Symbol	Explanation
$\boldsymbol{G(V,E)}$	Trust network
$\boldsymbol{v_s}$, $\boldsymbol{v_t}$	Source, target
$\boldsymbol{k}$	iteration number
$\boldsymbol{SP}$	Strength of path
$\boldsymbol{A_0}$	Root automaton corresponding to $\boldsymbol{v_s}$
$\boldsymbol{A_1}$	An automaton in the DLA
$\boldsymbol{A_t}$	Target automaton corresponding to v_t
$\boldsymbol{aLA_i}$	Action list of A_i

determine an increase and a decrease in the actions' probabilities. Based on learning parameters, three learning algorithms are introduced. (1) Linear reward-penalty algorithm (L_{R-P}) if $a = b$. (2) Linear reward-ε penalty algorithm ($L_{R-\varepsilon P}$) if $a \gg b$. (3) Linear reward-inaction algorithm (L_{R-I}) if $b = 0$, where the $\hat{p}(n)$ is constant if the chosen action receives a penalty from the environment. In general, L_{R-P} algorithms with $b \neq 0$ are ergodic schemes in which the automaton's action probability distribution $p(n)$ converges to a random variable p^* that is also independent of initial probability distribution $p(0)$ [24]. The algorithm can converge to various actions instead of converging to one optimal action. When it comes to online social networks, it means that the automaton is learning more reliable neighbors. Learning parameters influence the rate of convergence. If "a" is too small, the learning algorithm is too slow. Otherwise, if "a" is too large, action probabilities increase too fast [32]. As a result, the automaton's accuracy of finding optimal behavior decreases.

3.1. Distributed learning automata

A distributed learning automata (DLA) [34] is a network of interconnected learning automata that works together to solve a problem. In this network, only one automaton can be active at each instant n. The number of actions of each automaton is equal to the number of connected automata to it. Choosing an action by a LA activates another learning automaton corresponding to this action. Fig. 4 shows a DLA with two actions. If A_1 chooses the action α_{13}, then A_3 will be activated and selects one of its actions randomly, and consequently, one of the automata that is connected to A_3 is activated. Note that only one automaton can be active at each time.

A DLA is represented by a quadruple A, E, T, A_0 where, $A = A_1, \ldots, A_n$ is the set of learning automata, $E \subset A \times A$ indicates the set of edges, where edge e_{ij} is related to the action α_{ij} of the automaton A_i, T is the learning algorithm used to update the action probabilities vectors, and A_0 denotes the root automata from which activation process begins.

4. Proposed model

According to the presented studies and challenges, a new model is designed for calculating trust in this paper, and it contains the following three main steps:

Step 1. Preprocessing OSN to embed it into a network with a small scale.

Step 2. Building a trust network to generate a trusted graph (Section 4.1.2).

Step 3. Computing trust through the generated trusted graph. (Section 4.1.3)

A general framework for the proposed algorithm is depicted in Fig. 5.

These steps are virtually the same in graph-based approaches. What makes our proposed algorithm different is applying DLA to find trusted paths instead of using BFS in the trusted graph (Step 3). In Step 3, an intelligent algorithm that possesses the capability to discover trusted paths in a trust network without employing any restriction is proposed. Fig. 6 shows the flowchart of the proposed algorithm, which will be explained further in Section 4.1.3. The notations used in the flowchart are described in Table 1.

4.1. Problem formulation

An OSN is usually modeled by an acyclic weighted directed graph, $G = (V, E)$, where V denotes the set of nodes of the graph corresponding to OSN's users and $E \subseteq V \times V$ is the set of edges that correspond to the trust relations of users in OSN. The following definitions are commonly used in graph-based approaches.

Definition 1 (*Trustor*)**.** A user who intends to compute the trust level of another user is known as a trustor.

Definition 2 (*Trustee*)**.** A user whose trustworthiness is being calculated is known as a trustee.

Definition 3 (*Trust Network*)**.** A network in which its links represent the trust relations of users.

Definition 4 (*Trusted Graph*)**.** A subgraph of the trust network consisting of all existing trusted paths between a trustor and a trustee. Fig. 7 shows an example of a trust network.

Definition 5 (*Domain*)**.** A domain is a basic unit where users share some of their interests with each other. Domain should not be confused with the term "group" that is used in some other studies. In fact, the domain is related to the network's architecture, while the group is defined by OSN's users. Therefore, the number of attributes of a domain remains unchanged in the network while that of a group is usually changeable.

Definition 6 (*Active Domain*)**.** An active domain contains all the domains in which a user has participated.

Definition 7 (*Social Distance*)**.** The social distance from user i to user j is defined as:

$$Sd(i,j) = D_j - CD_{ij} + 1 \tag{8}$$

In Eq. (8) D_j represents the number of active domains of user j and CD_{ij} indicates the number of common domains of user i and its neighbor j.

4.1.1. Preprocessing an online social network

In the preprocessing, a user's neighbors are determined based on their social distance from the user. As mentioned above, social distance is the number of domains in which two users commonly

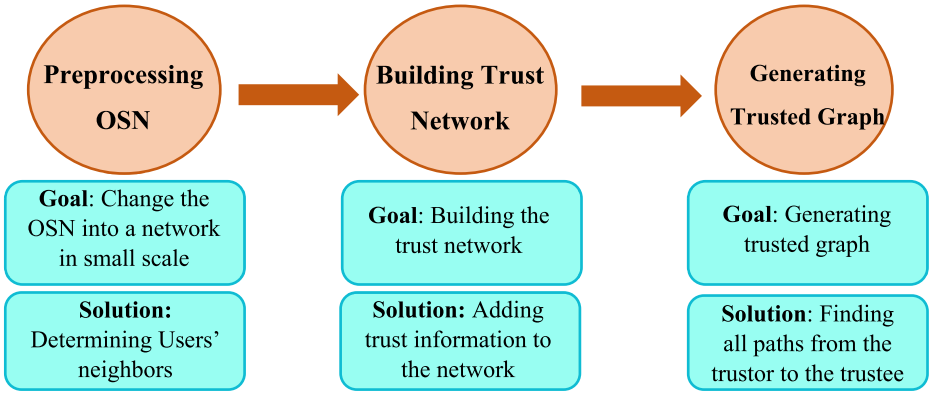


Fig. 5. The framework of the proposed algorithm.

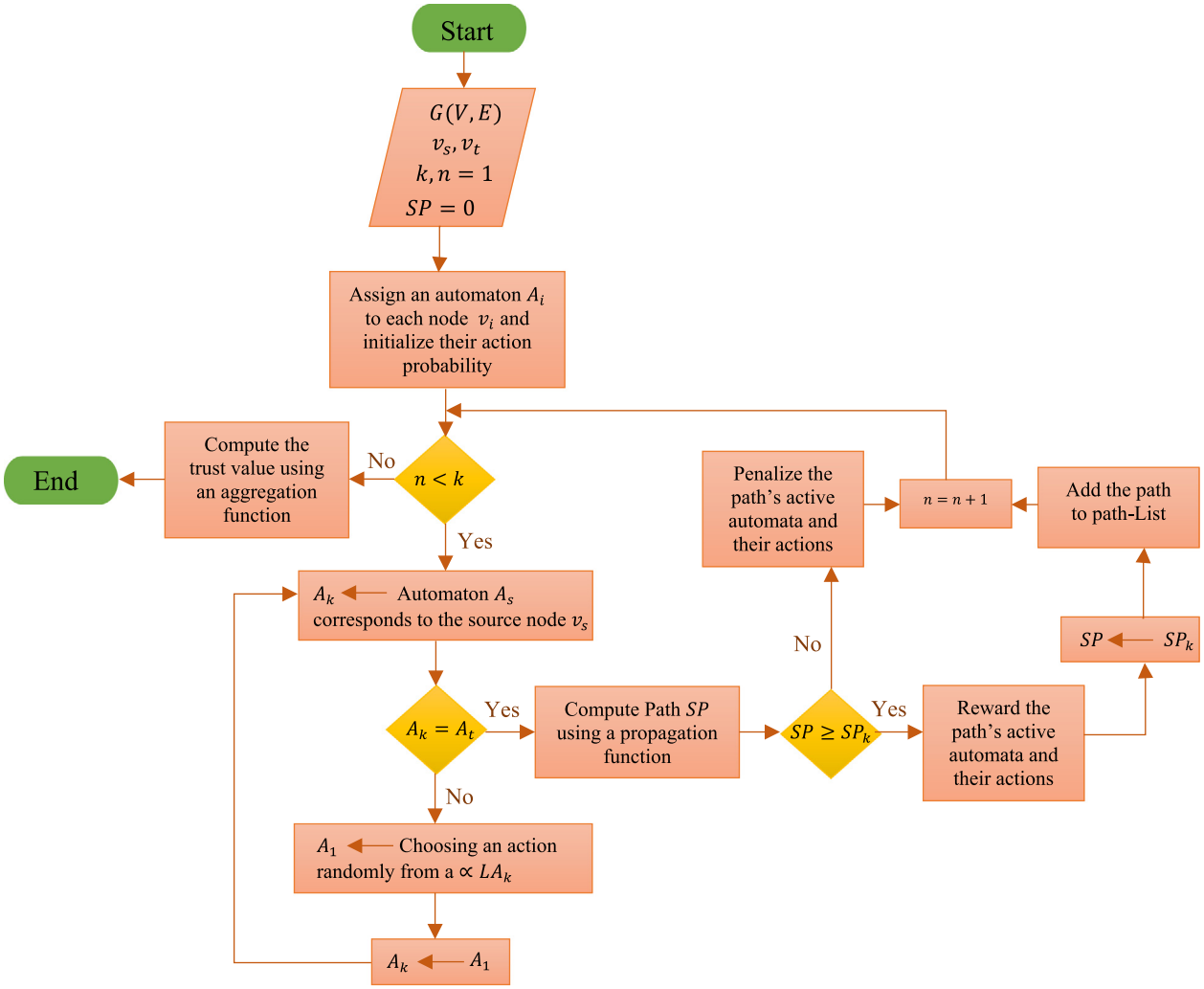


Fig. 6. The flowchart of Step 3 of the proposed algorithm.

participate. Therefore, a user's neighbors include those who appear in at least one identical domain as the user. Thus far, the social network is preprocessed and each user's neighbors are determined.

4.1.2. Building a trust network

In this step, trust information is added to the network. In other words, trust values are assigned to nodes by Richardson's technique [35] with a small modification ([36]). This technique uses the concept of a user's quality $q_i \in [0, 1]$ to assign trust values. A user's quality is a probability that the user's assertion about trustworthiness of other users is true (based on a normal distribution). For a pair of users i and j, trust is calculated by Richardson's technique:

$$T(i,j) = \text{uniformly chosen from}[\max(q_i - \delta_{ij}, 0), \min(q_i + \delta_{ij}, 1)] \quad (9)$$

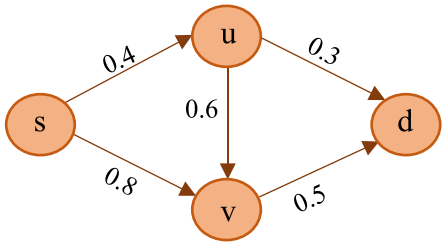


Fig. 7. An example of a trust network.

where $\delta_{ij} = (1 - q_i)/2$ is a noise parameter that determines how accurate users were at estimating the quality of the user they were trusting.

4.1.3. Generating a trusted graph

The next step is to generate trusted graphs. All paths from the trustor to the trustee are found using the LA.

At first, a DLA network is constructed by assigning a learning automaton to each node of the input network (trust network). The action set of each LA includes the neighbors of the node equipped with that LA. The activation process always begins from the automaton A_s (the root automaton) corresponding to the start node (trustor) v_s, and only one automaton can be active at each time.

For each LA, all actions have the same initial probability of being chosen. This initial probability is determined based on the scaled action probability vector $p\hat{}(n)$ (Eq. (4)).

Now, we have a DLA network incorporating LAs and their action list. To be able to generate a trusted graph (Definition 4), we need to find trusted paths between them. Therefore, the next step is to find the trusted paths from the trustor to the trustee. For this purpose, first, the root automaton A_s (related to the trustor node v_s) is activated and it selects one of its actions from its active actions set randomly for the next node along the trusted path. It is worth mentioning that the set of active actions of this automaton consists of v_s's neighbors which have not already been activated along the path by the automaton. If the active action set of the activated automaton is empty, the trusted path cannot continue forward and DLA may stop. Otherwise, A_s selects one of the available actions (v_k) according to the scaled action probability vector as shown in Eq. (4). Now, this selected action (v_k) is required to be checked: If v_k is our target, the trusted path between the trustor and the trustee is found and this step successfully is terminated. However, if (v_k) is not the trustee, automaton A_k (related to v_k) is activated and A_k selects one of its available actions. This procedure continues until a trusted path is found. Then, the strength of the trusted path is calculated using the propagation method. This method computes the strength of the trusted path from the trustor to the direct neighbor of the target node.

Two commonly used functions of the propagation method are Min and Multiply. Min takes the minimum trust value among all of the trust values along a trusted path. In contrast, Multiply computes the product of the trust values of all the referral trust values [11]. It is worth mentioning that the propagable property of trust makes it possible to compute the strength of the trusted path along a path. After determining the strength of the trusted path, it is compared to the reliability, which has already been obtained for the direct neighbor of the target. This reliability is based on recommendations of other nodes in a path. There is usually more than one path from a trustor to a trustee. Thus, the reliability of direct neighbors of the trustee needs to be computed for all paths from the trustor towards the trustee in the trusted graph. Besides, it is possible to have more than one path ending to a specific direct neighbor of the trustee. In this case, the strength of the new trusted path compares to the reliability of that direct neighbor, which is obtained from the previously found paths. If the strength of the trusted path is larger than or equal to the reliability of the direct neighbor, we update the reliability of the direct neighbor with the strength of the path, and the active learning automata along the trusted path receive rewards. That is, probability vectors of all active automata along the path and their probability vectors of active actions set update based on Eq. (5). Otherwise, they receive a penalty as shown in Eq. (6), and the probabilities of active automata along the path decrease. This activating process of DLA is performed again until the solution of the problem (finding reliable paths from the trustor to the trustee) for which DLA is designed is achieved. The LA learns that those nodes leading to the target are prior to those that do not lead to the target node. Each of the following conditions can be considered for the algorithm termination:

(1) Entropy calculation: Entropy shows the uncertainty of the system. In the case of learning automata, when it converges, entropy must decrease. In other words, the sum of the actions entropies of all automata decreases with time.
(2) Production of probabilities of active automata along the path must be equal or larger than $\mathcal{P}$, where $\mathcal{P}$ is the threshold for the probability.
(3) Number of iterations
(4) Reward
(5) The number of paths must be equal to or larger than $\mathcal{K}$, where $\mathcal{K}$ is the threshold for the paths.

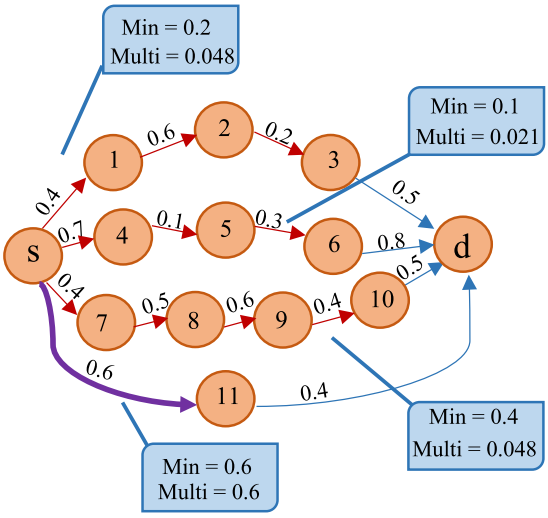
In this paper, to make a fair judgment between learning automata and breadth-first search as a deterministic algorithm, the number of iterations is taken as the algorithm termination condition. This criterion is the same in both algorithms.

Before starting the trust computation, the strength of the path acquired from the previous step is compared to the trust threshold $\mathcal{T}$, and if it is lower than $\mathcal{T}$, that path will be omitted.

The final predicted trust value is computed based on the weighted average of all functional trust values [30]. An example of trust computation using propagation and aggregation functions is shown in Fig. 8.

First, all paths from the source to the direct neighbors of the target are required to be found (Fig. 8, in red). Then, the reliability of the direct neighbors is calculated using Min or Multi functions. Min function takes the minimum value of all trust values along the paths and Multi-function multiplies all trust values in a path. The final trust value ($T(s, d)$) is computed using aggregation functions. Max function computes the final trust value based on the most reliable direct neighbor of the target. In Fig. 8, the most reliable direct neighbor is Node 11; therefore, the final trust value is 0.4 as the trust degree of this node to the target node. On the other hand, Wave calculates the weighted average of all functional trust values, which is equal to 0.4792 in Fig. 8.

Now, we have trust value from a source to a target. The reliability of the computation is actually the obtained accuracy. This accuracy generally shows the difference between the values is computed by the algorithm and the real value of trust we have from the dataset. The efficiency here in our algorithm is that we find more reliable paths from a source to a target since we do not remove paths blindly. In other algorithms (those with imposed restrictions) some reliable paths may be skipped just because they are too long or too short or due to other limitations. Here, learning automata, incorporating feedback interactions with the environment, learn nodes that end to the target and those which are reliable. The pseudo-code of the proposed algorithm is shown in Fig. 9.



$$T(s,d) = \begin{cases} 0.4 & \forall Aggregation: max \\ 0.4792 & \forall Aggregation: Wave \end{cases}$$

$$Aggregation: Wave$$

$$T(s,d) = \frac{(0.5 \times 0.2) + (0.8 \times 0.1) + (0.4 \times 0.5) + (0.6 \times 0.4)}{0.2 + 0.1 + 0.4 + 0.6} = 0.4792$$

Fig. 8. An example of trust computation.

5. Experiments

5.1. Experimental design

To evaluate the proposed algorithm, the standard leaves one out evaluation technique in machine learning is utilized. This method masks the direct link between the start node and the target node and calculates trust through the trusted graph. Then, the masked value is compared to the predicted trust value. Every value of experimental results is obtained by averaging all the 20 simulation runs. We also set the number of iterations to 700. This number of iterations is large enough to lead the LA to converge to the optimal action. Larger iteration number causes larger execution time, which is not desirable. In smaller iterations, the LA would not be given enough time to converge to the optimal action and thus the proposed algorithm would be a random solution rather than an intelligent one. The convergence rate of a learning automaton is shown in Fig. 10. From the figure, we can see that the penalty probability decreases with an increase in the number of iterations, and in 700 iterations the convergence rate becomes stable.

Two main metrics are considered: connection coverage and trust accuracy. Connection coverage shows the prediction power of the algorithm, and the number of predictable trust relations is determined. At least one trusted path is assumed to exist between the pair of nodes to be tested. Accuracy requires the algorithm to be able to determine whether a user is reliable or not. To test the coverage of the algorithm, the number of trust relations obtained from the algorithm to the total available trust relations is computed, and the result is considered as the connection coverage. To test the trust accuracy, trust metrics are used (see Table 2). These metrics are Fscore, recall, precision, and mean absolute error. Precision indicates the fraction of users that are predicted as reliable and are very reliable. Recall shows the fraction of users that are reliable and are successfully predicted. Fscore is the metric combining both recall and precision to introduce a new joint metric.

In Table 2, B_t represents the number of users that trust another user according to the estimated trust calculated with the algorithm, and A_t shows the number of pair nodes that directly trust each other.

Table 2
Evaluation metrics.

Metric	Equation
Error	$\lvert trustcalculated - actualtrust \rvert$
Precision	$Precision = (A_t \cap B_t)/B_t$
Recall	$Recall = (A_t \cap B_t)/A_t$
Fscore	$2 \times \frac{(Precision \times Recall)}{(precision + Recall)}$

5.2. Dataset

Neighbors are determined by the social distance from each user, which is based on the domain concept. Therefore, the applicable dataset needs to include the domain information of each user. Subsequently, if defining domain information for the content of an online social network becomes difficult, this algorithm will not be a proper choice to handle the network. However, it is possible to implement data mining techniques on such a network to classify the content into different categories, and hence the proposed algorithm may be utilized to compute trust relations between users in the OSNs.

The dataset from Epinions.com [13] is used here. This dataset contains 717,667 trust statements stated by 132,000 users, 1,156,144 articles and 13,668,319 article ratings. Both users and items are displayed by unidentified numerical signs. Epinions is a well-known online product review website where members can write reviews about products and rate the other members' reviews. To get trust information, a file is created that includes three columns. The first column contains the ID of users who make trust statements about the other users, the second column contains the ID of users whose trustworthiness is evaluated, and the third column covers the trust information with value 1. Obtaining domain information is not as easy as getting trust relations, and it needs the dataset being processed. For this purpose, it is assumed that those goods that can be categorized in the same group are in the same domain. The 1,560,144 articles are sorted in ascending order, and the first 50,000 are made into a smaller dataset. 3142 articles of these 50,000 are un-duplicated, and they are placed in 6 domains. From the 20,075 unique article authors, the first 5000 are chosen, and their domain information is collected. Finally, IDs not included in 5000 article authors are deleted from the trust analysis file and the remaining 51,888 edges and 3169 nodes are used for the experiments. Table 3 illustrates a portion of Epinions dataset including nodes and domains information. The first column shows the nodes number in an anonymous numerical format. The other columns include the number of specific domains in which each node participated. For instance, the node with a number of 241578 participated in domains 3, 4 and 5.

Algorithm: proposed algorithm for trust evaluation

Input: Trust network $\boldsymbol{G(V,E)}$, Maximum number of iterations $\boldsymbol{K}$, Source nodes $\boldsymbol{v_s}$, Target nodes

Output: Trust value

Assumptions

Assign an automaton $\boldsymbol{A_i}$ to each node $\boldsymbol{v_i}$ and initialize their action probability

Let k denotes the number of iterations that initially set to $\mathbf{1}$

Set the Strength of path $\boldsymbol{SP_i}$ to 0

Begin

While (k < $\boldsymbol{K}$)

$\boldsymbol{A_k}$ ← Automaton $\boldsymbol{A_s}$ corresponding to the source node $\boldsymbol{v_s}$

Path includes the traversed nodes along the path from $\boldsymbol{v_s}$ to $\boldsymbol{v_t}$ which initially set to $\boldsymbol{v_s}$

While (there is no available action or $\boldsymbol{v_k} \in \boldsymbol{aLA_t}$)

Select node $\boldsymbol{v_s}$ randomly as a starting node

If (the action set is not empty) **then**

$\boldsymbol{v_k}$ ← Automaton $\boldsymbol{A_k}$ is active and then selects an action according to its action probability vectors

Path ← Path ∪ $\boldsymbol{v_k}$

End If

End While

$\boldsymbol{SP}$ ← Compute the strengthen of path using a propagation function

If ($\boldsymbol{v_k} \in \boldsymbol{aLA_t}$ and $\boldsymbol{SP} \geq \boldsymbol{SP_k}$) **then** // favorable traverse

Reward all active automata of the path and their actions

$\boldsymbol{SP_k}$ ← $\boldsymbol{SP}$

k ← k + 1

Else If

Penalize the path's all active automaton and their actions

k ← k + 1

End While

Compute the trust value using an aggregation function

End Algorithm

Fig. 9. Pseudo-code of the proposed algorithm.

Table 3
A portion of the dataset used for trust evaluation.

Node No.	Domain					
206437	4	5	6			
216430	1	5	6			
223652	1	6				
243427	1	2	3	4	5	6
234885	3	4	5	6		
241578	3	4	5			

5.3. Assigning trust values

Assigning trust values to the nodes of the network is done using the Richardson technique [35]. Table 4 shows the accuracy parameters for experiments. *grate* and *brate* variables show the quality of nodes (good or bad) in Richardson's technique, a demonstration of population percentages of good nodes and bad nodes which corresponds to the quality of nodes, and trust values will be defined by the quality. Furthermore, user qualities are choosing from two normal distributions, good nodes ($\mu = 0.75$) and bad nodes ($\mu = 0.25$) with the same standard deviation for both of them ($\sigma = 0.25$). In our dataset, 30% of the total nodes are bad and 70% are good.

5.4. Simulation results

To investigate the efficiency of the proposed algorithm, four strategies shown in Table 5 are implemented and their performances are compared. These strategies are commonly used in the

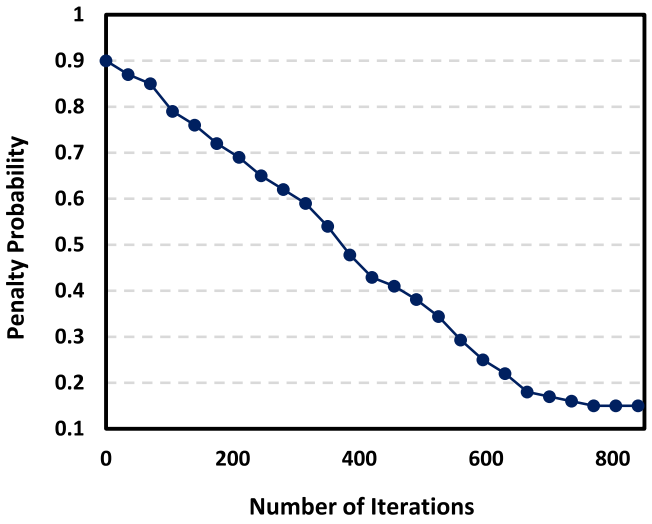


Fig. 10. Convergence rate of a learning automaton.

Table 4
Experiment parameters.

Parameter	Definition	Default value
μ	Mean of normal distribution	
σ	Standard deviation	%25
brate	% bad nodes	%30
grate	% good nodes	%70
$\mathcal{T}$	Trust threshold	0.4

Table 5
The common strategies for trust evaluation.

Strategy	Propagation	Aggregation
Min-Max	Min	Max
Min-Wave	Min	Wave
Multi-Max	Multi	Max
Multi-Wave	Multi	Multi

literature to evaluate trust [1]. Several experiments are carried out on the real dataset of Epinions to evaluate the performance of the proposed algorithm. The proposed algorithm is compared with SWTrust, TidalTrust, and MoleTrust algorithms, which are well-established graph-based algorithms.

5.4.1. Experiment 1: Effects of learning parameters

This experiment seeks to compare the performance of linear learning algorithm $L_{R-\varepsilon P}$ with L_{R-P}. The comparison is made in terms of accuracy when the learning parameters a and b are varied. The results are summarized in Table 6 for $L_{R-\varepsilon P}$ and Table 5 for L_{R-P}. We observe that small values for learning parameters result in lower MAE. On the other hand, higher learning parameters lead to non-optimal convergences in the first few iterations. However, as the learning parameter becomes smaller, the proposed algorithm takes enough time to converge to the optimal actions after running a set of iterations based on the feedback relation between automaton and environment. Therefore, trust accuracy is very sensitive to the value of the learning parameter a. Large (small) value of a results in lower (higher) accuracy.

Tables 6 and 7 show that MAE values for L_{R-P} are higher than those for $L_{R-\varepsilon P}$. In other words, L_{R-P} converges faster, and thus obtainable trust relations from the chains of friends decrease. This means fewer opinions are available for inferring trust, which results in one-sided trust.

In the following experiments, the learning algorithm $L_{R-\varepsilon P}$ with learning parameters $a = 0.009$ and $b = 0.00009$ is used.

Table 6
Effect of the learning parameters in $L_{R-\varepsilon P}$.

a	b	Recall	Precision	Fscore	MAE
0.001	0.00001	0.9685	0.8815	0.9229	0.1138
0.005	0.00005	0.9278	0.9158	0.9217	0.1140
0.009	0.00009	0.9526	0.9374	0.9449	0.1158
0.01	0.0001	0.9356	0.9191	0.9272	0.1162
0.07	0.0007	0.9234	0.8968	0.9092	0.1172
0.1	0.001	0.8995	0.8984	0.8989	0.1184

Table 7
Effect of the learning parameters in L_{R-P}.

a	b	Recall	Precision	Fscore	MAE
0.001	0.00001	0.9495	0.9015	0.9248	0.1221
0.005	0.00005	0.9326	0.9264	0.9294	0.1225
0.009	0.00009	0.9246	0.9094	0.9169	0.1134
0.01	0.0001	0.9179	0.8986	0.9081	0.1238
0.07	0.0007	0.9153	0.8841	0.8994	0.1252
0.1	0.001	0.9098	0.8784	0.8938	0.1161

Table 8
Types of relations.

Weak ties	Strong ties
60.78%	39.22%

5.4.2. Experiment 2: The strength of weak ties

This experiment is carried out to verify the validness of "the strength of weak ties" theory in our proposed algorithm. This theory is derived from Nick Granovetter's study about the spread of information through social networks [37]. In this article, it is proven that weak ties contain precise information and are very valuable relations. In the proposed algorithm, the process of choosing neighbors for the next hop along a path toward the target is done randomly. In order to validate the "the strength of weak ties" theory, in each path $(n_1, \ldots, n_i, n_{i+1}, \ldots, n_k)$, social distance from n_{i+1} to n_i is computed. Then, if the social distance is equal to 1 ($d(i, ij) = 1$), the relation is taken as a strong tie, and if $d(i, ij) > 1$, it is considered a weak tie. The result is shown in Table 8. From the results, we observe that 60.78% of all trust relations are weak ties and only 39.22% of trust relations are strong ties. We can conclude from this observation that trusted paths are mostly made of weak ties which verifies the validity of "the strength of weak ties" theory in our proposed algorithm.

5.4.3. Experiment 3: Coverage rate

This experiment aims to compare the coverage rate of the proposed algorithm with breadth-first search-based algorithms (TidalTrust, MoleTrust, SWTrust, and the algorithm in [26], which we call CombinedTrused in the rest of the paper for simplicity). To perform this experiment, we plot the coverage with respect to the max length. Maximum path length varies in the range of [2, 10].

As seen in Fig. 11, the coverage increases with the length. Furthermore, algorithms reach maximum coverage when the max length is 6, 8, and 10. What we can conclude from this observation is that if a trusted path exists between two users, its length will not be excessively long. The figure also shows that the max coverage of proposed algorithm is about 0.95, which has a 10% improvement in comparison with SWTrust (as the worst coverage) and 5% with CombinedTrust network. The reason for this improvement is that using BFS in a large-scale network such as an online social network is subject to restrictions implementation on the searching process due to computation complexity. Therefore, some paths are removed blindly and some trust relations become unreachable. Losing reliable relations contributes to decreasing the coverage. On the contrary, using the learning automata for

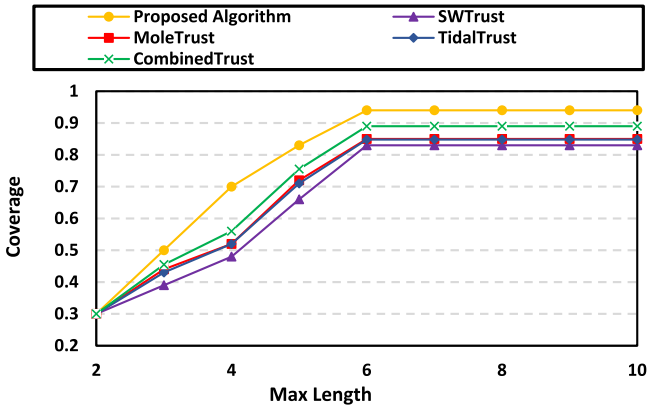


Fig. 11. Coverage rate.

Table 9
The comparison between different strategies.

Strategy	Recall	Precision	Fscore	MAE
Min-Max	0.9259	0.9245	0.9251	0.1268
Min-Wave	0.9587	0.9217	0.9328	0.1248
Multi-Max	0.9295	0.9268	0.9281	0.1252
Multi-Wave	0.9388	0.9317	0.9352	0.1270

finding trusted paths between users in a social network does not require setting limitations; Learning automata learns unreliable users through feedback interaction with the environment and subsequently removes only untrustworthy paths. Thus, reliable relations will still contribute to inferring trust, and the coverage rate will not decrease in comparison with other algorithms.

5.4.4. Experiment 4: Effect of common evaluating strategies

This experiment is conducted to compare the effect of different strategy implementations (Table 4) on the efficiency of the proposed algorithm. The comparison is made in terms of accuracy metrics, and the results are reported in Table 9. From the results, we can see that accuracy remains high with the minimum Fscore of 0.9251. This value verifies that the proposed algorithm is a high-performance algorithm. We also observe that use of Min or Multi functions has the same contribution in terms of accuracy since using these functions makes no considerable changes in Fscore values (ex. Min-Wave and Multi-Wave values have a scanty difference of 0.0024). On the contrary, using Max or Wave functions causes remarkable changes in the accuracy values (ex. Min-Max and Min-Wave values difference is equal to 0.0076). The accuracy of using Wave is higher than that of using Max. The reason is that the former considers all opinions obtained from multiple paths in trust calculation while Max uses only one opinion. One opinion may include a degree of bias; however, considering average of all opinions leads to a more objective result. As a result, a more unbiased trust leads to higher accuracy. Based on the accuracy consideration, for the remaining experiments, the results of the Min-Wave strategy were reported.

5.4.5. Experiment 5: Efficiency comparison

This experiment is performed to compare the efficiency of the proposed algorithm with some other trust graph-based algorithms in terms of accuracy metrics. The algorithms for comparison are TidalTrust, MoleTrust, SWTrust, and CombinedTrust. Results are averaged on 20 simulation runs with a 95% CI [0.9168,0.9598] and are given in Table 10.

As shown by the results in Table 10, the accuracy of our proposed algorithm has reached 0.9398. This accuracy is significant because of the following findings.

Table 10
Comparison of different algorithms in terms of accuracy.

Model	Recall	Precision	Fscore (95% CI)	MAE	Time
TidalTrust	0.8897	0.8654	[0.8647, 0.8798]	0.1298	2.0841
MoleTrust	0.8927	0.8632	[0.8492, 0.8816]	0.1301	2.6184
SWTrust	0.8460	0.8413	[0.8224, 0.8459]	0.1278	7.1278
CombinedTrust	0.8994	0.8683	[0.8762, 0.8946]	0.1287	3.1245
Proposed-algorithm	0.9587	0.9217	[0.9257, 0.9598]	0.1248	13.4823

(1) The proposed algorithm has an improvement of 6% in performance compared to the other models, with the setting of no limitation on the search process to find trusted paths between trustor and trustee. Specifically, other algorithms need to put restrictions on the search process: in TidalTrust only the shortest and strongest paths are considered; in MoleTrust, the number of nodes from the source node is restricted; and in SWTrust, as well as CombinedTrust, the width of search is restricted. The proposed algorithm finds trusted paths without any restriction, and hence no paths will be omitted unconsciously and all trust relations are reachable. This is one important feature of learning algorithms, based on which we not only do not remove any relation blindly but also use feedback connections to learn what would happen next based on the former happenings and then with a degree of probability choose the best solution (here, the reliable path).

(2) All other algorithms' maximum, minimum, and mean values of accuracy are between 0.8 and 0.9. In contrast, the minimum accuracy value of the proposed algorithm in this paper is 0.9257, showing better reliability in comparison with the other algorithms. Furthermore, this is noticeable because we have rarely seen an algorithm with an accuracy more than 0.9 in graph-based algorithms.

However, the cost is the higher calculation time. While we aim to achieve an algorithm that improves both time efficiency and precision, for many well-developed algorithms, it is sometimes not possible to simultaneously improve both. Namely, when one is improved, the other is not or slightly lowered as a compensation. For our algorithm, accuracy is improved and time efficiency is slightly lowered. The algorithm takes about 13 s to execute, which is more than the other algorithms. This is because of the learning nature of the proposed algorithm. That is, the learning automata needs more time to converge to the action with high probability; otherwise, it would be a random solution and would not show over-performance compared to unintelligent algorithms.

Furthermore, the proposed algorithm does not impose any constraints and traverses all paths between trustee and trustor; however, the other algorithms require setting restrictions on the search process and thus do not traverse all existing paths (in TidalTrust, only the shortest and strongest paths are considered; in MoleTrust, the number of nodes from the source node is restricted; and in SWTrust, the width of search is limited). Consequently, they require less time compared to the proposed algorithm.

5.4.6. Experiment 6: the effect of max length

In this experiment, we study the effect of the max length on the performance of the proposed algorithm compared to TidalTrust, MoleTrust, SWTrust and CombinedTrust with an average margin of error of 1.8%. For this purpose, the max length was changed from 2 to 8, and the trust threshold was set to 0.4. The mean error and Fscore with respect to the max length are shown in Fig. 12 (a, c) and (b).

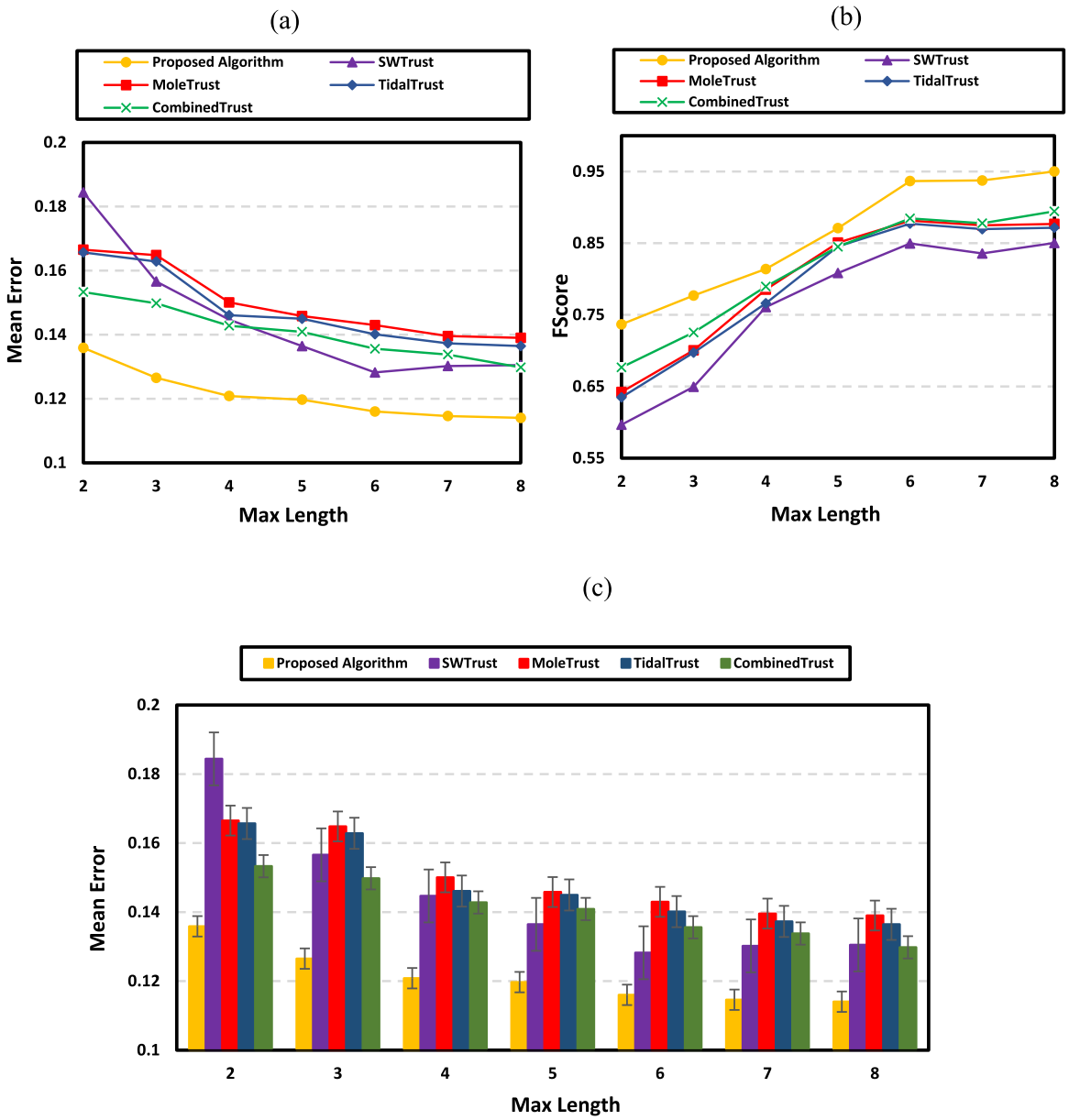


Fig. 12. The effect of max length on (a, c) mean error and (b) Fscore.

We can see from Fig. 12(b) that increasing max length results in higher accuracy. When max length is equal to 2, the accuracy attains its minimum value because fewer paths with short lengths exist, which leads to fewer available evidence for trust computation and, therefore, the accuracy decreases. In the lower lengths (2 to 6), trust increases dramatically. The reason is that in lower lengths, the number of paths increases significantly, providing more information for trust inference; however, fewer new paths are obtained in larger lengths. Consequently, the evidence for inferring trust decays, and as a result, accuracy augmentation is nominal compared to that of low lengths. Furthermore, we can see from Fig. 12(b) that for TidalTrust, MoleTrust, SWTrust, and CombinedTrust in large max lengths (6 to 8), the accuracy drops a little; specifically, there is a little decrease in trust accuracy when max length = 7. The reason is that increasing the number of paths contributes to an increase in the number of paths with lower trust. These paths with lower trust cause the trust accuracy to decrease. Nevertheless, in the proposed algorithm, by increasing the max length, the accuracy of trust computation does not decrease. Increasing the amount of evidence and choosing reliable paths are the reasons for this phenomenon. Fig. 12(a, c) shows that increasing the length of paths results in a lower mean error. The reason is that searching with a larger max length leads to more available paths which increases the amount of available evidence. Therefore, decreasing the difference between the real trust values and those calculated through the proposed algorithm causes relative error to decrease.

5.4.7. *Experiment 7: Test the "trust threshold" influence*

This experiment is carried out to study the impact of varying trust threshold rates on the performance of the proposed algorithm compared to TidalTrust, MoleTrust, SWTrust, and CombinedTrust algorithms in terms of accuracy metrics—the average of margin of errors is equal to 1.9%. The trust threshold varies in the range of [0.1, 0.9], and the max length is set to 6. The mean error and Fscore are plotted concerning the trust threshold in Fig. 13(a, c) and (b).

From Fig. 13(a, c) we conclude that increasing the trust threshold results in decreased accuracy, which is more pronounced

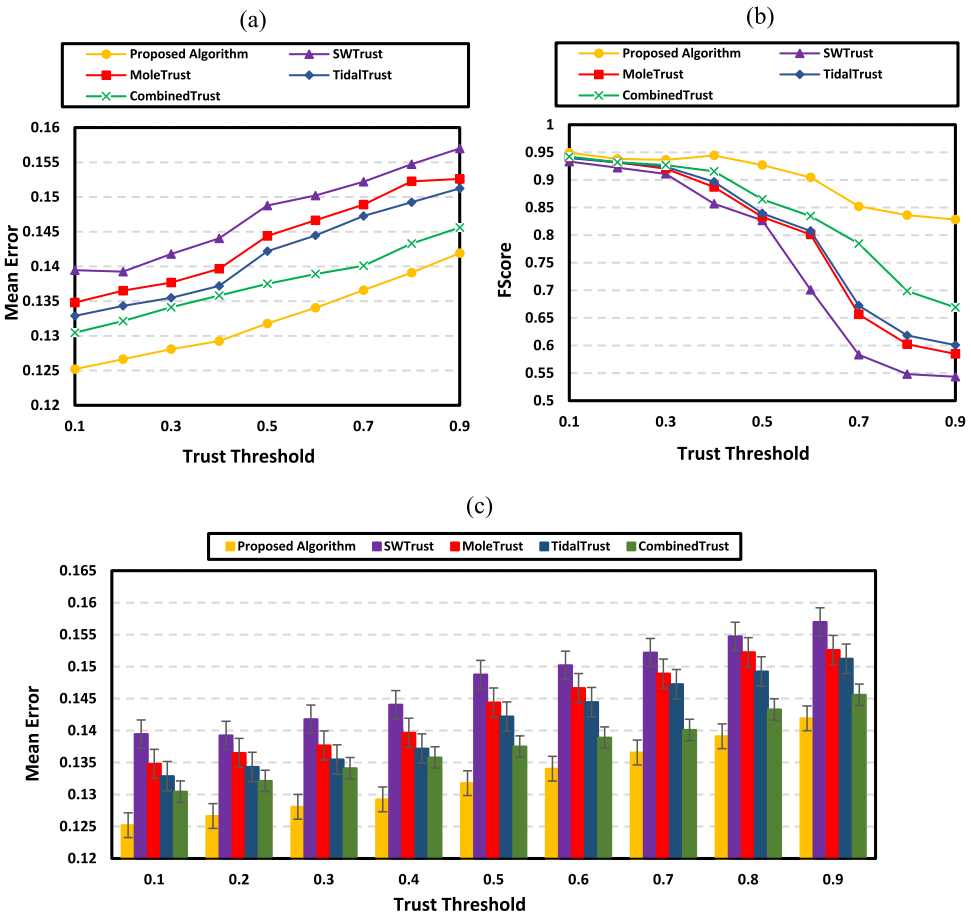


Fig. 13. The effect of trust threshold on (a, c) mean error and (b) Fscore.

when threshold = 0.6. When the trust threshold is 0.6, the trust accuracy and mean error sharply decreases and increases respectively because a higher threshold means more untrusted paths exist, and thus more paths are removed. In this way, fewer paths are available for trust computation, resulting in fewer available opinions, leading to more subjective and one-sided final trust estimation. Therefore, we observe a sharp decrease in accuracy in higher trust threshold. In conclusion, trust threshold should not be set too high. On the other hand, in small values of trust threshold thus less restriction, most paths are regarded as trustful and used for trust computation. Consequently, more evidence is available for trust computation, and trust accuracy increases.

5.4.8. Experiment 8: the effect of the number of direct neighbors

The experiment aims to investigate the impact of the number of direct neighbors of the target node on trust accuracy. The Fscore and mean error for the proposed algorithm and the other algorithms TidalTrust, MoleTrust, SWTrust, and CombinedTrust are plotted in Fig. 14.

We can see from Fig. 14 that an increase in the number of direct neighbors causes trust accuracy to increase. This is because direct neighbors of the target are responsible for functional trust values, which directly are put in aggregation functions' equation for trust computation. Therefore, their weight in trust evaluation is significant. More direct neighbors mean more available opinions leading to higher trust accuracy. Moreover, because the proposed algorithm uses all paths for trust computation, according to results, it has a higher accuracy in comparison with other algorithms.

6. Conclusion and future works

In this paper, we investigated the trust evaluation in online social networks. Traditional graph-based approaches find the trusted paths between two users by restricting the breadth-first search. As a result of the limitation, some trust relations become unreachable, which leads to decreased trust accuracy and coverage.

In order to overcome this restriction, we proposed an algorithm that is a combination of the graph-based algorithm and an artificial intelligence-based algorithm. The graph-based approach calculates the trust by generating small trusted graphs, and each node of the network is equipped with a learning automaton. Then, the trusted paths between two users are found by using distributed learning automata.

In order to determine the efficiency of the proposed algorithm, it was compared to other existing graph-based models, and trust was evaluated by applying limitations on the BFS. The result showed that the accuracy of proposed algorithm reached to 0.9398, which is a 6% improvement comparable to the existing algorithms. However, as a compensation, it is a time-consuming algorithm, consuming 13.4823 s, which is at least twice as much time as used by some existing other algorithms.

In future works, our aim will be to decrease the executable time. For this purpose, we plan to design a new hybrid model to combine both the BFS and the learning automata incorporating the Dempster–Shafer evidential theory, Choquet integral and Quantum probability theory. In this way, the breadth-first search, which is a deterministic algorithm, will become an intelligence algorithm.

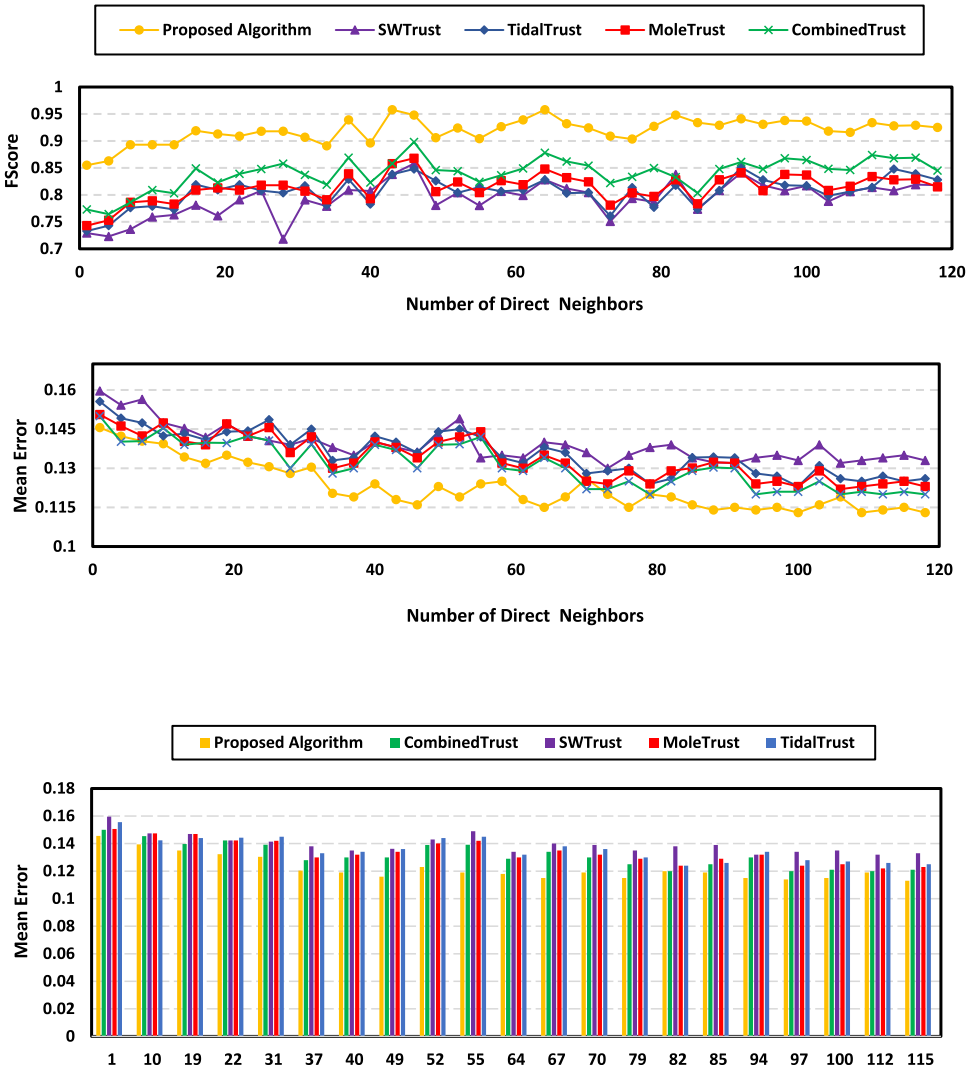


Fig. 14. The effect of the number of direct neighbors.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was support by the Ministry of Science and Technology under Grant MOST 106-2410-H-182 -004 and Chang Gung Medical Foundation under Grant BMRPA79.

References

[1] W. Jiang, G. Wang, J. Wu, Generating trusted graphs for trust evaluation in online social networks, Future Gener. Comput. Syst. 31 (2014) 48–58.
[2] N. Fatehi, H. Shahhoseini, A hybrid algorithm for evaluating trust in online social networks, in: 2020 10th International Conference on Computer and Knowledge Engineering, ICCKE, 2020, pp. 158–162.
[3] L. Jiang, Y. Cheng, L. Yang, J. Li, H. Yan, X. Wang, A trust-based collaborative filtering algorithm for E-commerce recommendation system, J. Ambient Intell. Humaniz. Comput. 10 (2019) 3023–3034.
[4] Z. Gong, W. Guo, E. Herrera-Viedma, Z. Gong, G. Wei, Consistency and consensus modeling of linear uncertain preference relations, European J. Oper. Res. 283 (2020) 290–307.
[5] M. Cai, L. Yan, Z. Gong, G. Wei, A voting mechanism designed for talent shows in mass media: Weighted preference of group decision makers in social networks using fuzzy measures and choquet integral, Group Decis. Negot. (2020) 1–24.
[6] U.E. Tahta, S. Sen, A.B. Can, GenTrust: A genetic trust management model for peer-to-peer systems, Appl. Soft Comput. 34 (2015) 693–704.
[7] T. Wang, H. Luo, W. Jia, A. Liu, M. Xie, MTES: An intelligent trust evaluation scheme in sensor-cloud enabled industrial internet of things, IEEE Trans. Ind. Inf. (2019).
[8] Z. Zhang, J. Wen, X. Wang, C. Zhao, A novel crowd evaluation method for security and trustworthiness of online social networks platforms based on signaling theory, J. Comput. Sci. 26 (2018) 468–477.
[9] W. Lin, Z. Gao, B. Li, Guardian: Evaluating trust in online social networks with graph convolutional networks, in: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, 2020, pp. 914–923.
[10] J.A. Golbeck, Computing and Applying Trust in Web-Based Social Networks (Ph.D. thesis), University of Maryland, 2005.
[11] W. Jiang, G. Wang, M.Z.A. Bhuiyan, J. Wu, Understanding graph-based trust evaluation in online social networks: Methodologies and challenges, ACM Comput. Surv. 49 (2016) 10.
[12] A. Jøsang, R. Ismail, C. Boyd, A survey of trust and reputation systems for online service provision, Decis. Support Syst. 43 (2007) 618–644.
[13] P. Massa, P. Avesani, Trust metrics on controversial users: Balancing between tyranny of the majority, Int. J. Semant. Web Inform. Syst. (IJSWIS) 3 (2007) 39–64.
[14] W. Jiang, J. Wu, F. Li, G. Wang, H. Zheng, Trust evaluation in online social networks using generalized network flow, IEEE Trans. Comput. 65 (2015) 952–963.

[15] Z. Gong, H. Wang, W. Guo, Z. Gong, G. Wei, Measuring trust in social networks based on linear uncertainty theory, Inform. Sci. 508 (2020) 154–172.
[16] G. Xu, B. Liu, L. Jiao, X. Li, M. Feng, K. Liang, et al., Trust2Privacy: a novel fuzzy trust-to-privacy mechanism for mobile social networks, IEEE Wirel. Commun. 27 (2020) 72–78.
[17] J. Guo, A. Liu, K. Ota, M. Dong, X. Deng, N. Xiong, ITCN: an intelligent trust collaboration network system in IoT, IEEE Trans. Netw. Sci. Eng. (2021).
[18] X. Chen, Y. Yuan, M.A. Orgun, L. Lu, A topic-sensitive trust evaluation approach for users in online communities, Knowl.-Based Syst. (2020) 105546.
[19] R. Ureña, F. Chiclana, E. Herrera-Viedma, DeciTrustNET: A graph based trust and reputation framework for social networks, Inf. Fusion (2020).
[20] X. Zhuang, X. Tong, A local trust inferring algorithm based on reinforcement learning DoubleDQN in online social networks, in: 2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics, CISP-BMEI, 2020, pp. 1064–1069.
[21] G. Wang, J. Wu, FlowTrust: Trust inference with network flows, Front. Comput. Sci. China 5 (2011) 181.
[22] J. Golbeck, J. Hendler, Inferring binary trust relationships in web-based social networks, ACM Trans. Internet Technol. (TOIT) 6 (2006) 497–529.
[23] J.-P. Huang, B. Heidergott, I. Lindner, Naïve learning in social networks with random communication, Social Networks 58 (2019) 1–11.
[24] K.S. Narendra, M.A. Thathachar, Learning Automata: An Introduction, Courier Corporation, 2012.
[25] G. Wang, J. Wu, Multi-dimensional evidence-based trust management with multi-trusted paths, Future Gener. Comput. Syst. 27 (2011) 529–538.
[26] Y.A. Kim, An enhanced trust propagation approach with expertise and homophily-based trust networks, Knowl.-Based Syst. 82 (2015) 20–28.
[27] A. Jøsang, E. Gray, M. Kinateder, Simplification and analysis of transitive trust networks, Web Intell. Agent Syst. Int. J. 4 (2006) 139–161.
[28] M. Lesani, N. Montazeri, Fuzzy trust aggregation and personalized trust inference in virtual social networks, Comput. Intell. 25 (2009) 51–83.
[29] 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.
[30] 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.
[31] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, MIT Press, 2009.
[32] A. Rezvanian, B. Moradabadi, M. Ghavipour, M.M.D. Khomami, M.R. Meybodi, Introduction to learning automata models, in: Learning Automata Approach for Social Networks, Springer, 2019, pp. 1–49.
[33] H. Beigy, M.R. Meybodi, A learning automata-based algorithm for determination of the number of hidden units for three-layer neural networks, Internat. J. Systems Sci. 40 (2009) 101–118.
[34] M. Ghavipour, M.R. Meybodi, Trust propagation algorithm based on learning automata for inferring local trust in online social networks, Knowl.-Based Syst. 143 (2018) 307–316.
[35] M. Richardson, R. Agrawal, P. Domingos, Trust management for the semantic web, in: International Semantic Web Conference, 2003, pp. 351–368.
[36] S. Shekarpour, S. Katebi, Modeling and evaluation of trust with an extension in semantic web, Web Semant. Sci. Serv. Ag. World Wide Web 8 (2010) 26–36.
[37] M.S. Granovetter, The strength of weak ties, in: Social Networks, Elsevier, 1977, pp. 347–367.