Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 536717, 16 pages
http://dx.doi.org/10.1155/2014/536717

Research Article

Research on Trust Propagation Models in Reputation Management Systems

Zhiyuan Su,[1] Mingchu Li,[1] Xinxin Fan,[1] Xing Jin,[1] and Zhen Wang[1,2]

[1] *School of Software, Dalian University of Technology, Liaoning 116024, China*
[2] *Computational Social Science Lab, School of Innovation Experiment, Dalian University of Technology, Liaoning 116024, China*

Correspondence should be addressed to Zhen Wang; wangz@dlut.edu.cn

Received 11 March 2014; Accepted 9 May 2014; Published 28 May 2014

Academic Editor: Matjaz Perc

Copyright © 2014 Zhiyuan Su et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Feedback based reputation systems continue to gain popularity in eCommerce and social media systems today and reputation management in large social networks needs to manage cold start and sparseness in terms of feedback. Trust propagation has been widely recognized as an effective mechanism to handle these problems. In this paper we study the characterization of trust propagation models in the context of attack resilience. We characterize trust propagation models along three dimensions: (i) uniform propagation and conditional propagation, (ii) jump strategies for breaking unwanted cliques, and (iii) decay factors for differentiating recent trust history from remote past history. We formally and experimentally show that feedback similarity is a critical measure for countering colluding attacks in reputation systems. Without feedback similarity guided control, trust propagations are vulnerable to different types of colluding attacks.

1. Introduction

Large scale social networks provide such a convenient opportunity that has never been known before for people interacting with each other without face-to-face interaction, such as Facebook, eBay, and Epinion. Due to the features of behaviors in these networks, such as anonymity, dynamics, scalability, normal interactions are seriously threatened by dishonorable users. For example, a malicious seller may try to provide fake or low quality goods to a buyer in eBay for illegal profit or a dishonest user in Facebook may cheat another user for slandering someone else. Fortunately, there are a lot of ways to provide guidance for people identifying these malicious users or behaviors. Trust and reputation management systems are considered as one of the most effective methods to maintain these social networks.

In most cases, the interactions in social network could be seen as the request-respond mode. When sending an interacting request in a large scale social network, users usually have more than one responder. A list of users is offered to the requestor as responders so that the requestor could select someone as his/her final interaction partner. Reputation management provides a good opportunity for helping users selecting the trustworthy user from the responding list. Amazon, Epinion, and numerous other distributed social network systems have demonstrated that incorporating reputation management can be an effective way to improve the trustworthiness of the system. Trust computing, the core part of reputation management system, also has a wide range of applications in eCommerce, mobile computing, peer to peer computing, sensor computing, and social computing, to name a few [1–3]. Intuitively, computing trust value of a given user requires some indicators such as pretransaction based feedback information or something else. By feedback we mean the performance of all historical transactions of the target user. This can be expressed according to the rating process used in several sites, such as eBay [4], or a statement of trust as in the case of Epinion.

However, feedback based trust computing is still suffering some challenges in the following aspects. (i) Dishonest feedbacks: malicious users may slander good users in the system by giving relatively low rating values or exaggerating another bad user by giving relatively high rating values. (ii) Malicious collective or manipulation: malicious users could form a malicious collective to help improving each other's trust value. (iii) Sparsity problem: apparently, no matter what exact form the direct relationship takes, one user could only feature in direct relationships with a relatively small number of others in a feedback based trust system. Consequently, most of the responding users in the system would be strangers for a specific user when sending a transaction request. (iv) Cold start problem: basically, newcomers without any feedback from any other existing users in a trust system could hardly be selected as a trusted party. Specifically, it concerns the issue that the trust system cannot draw any inferences for users or items about which it has not yet gathered sufficient feedback information.

Lots of trust models [5–8] have been proposed to date the problems mentioned above. They all differ from one another in three aspects, including feedback aggregating algorithm, local trust, and global trust computing. But they are similar in one aspect; all are based on the uniform trust propagation kernel when dealing the sparsity and cold start problem. For example, EigenTrust [5] is through the use of Eigenvector based uniform propagation kernel and ServiceTrust [8] is based on a similarity weighted uniform propagation kernel.

Unfortunately, most of the existing approaches have been developed independently and little efforts have been made to compare and understand the relative strength and inherent vulnerabilities of different propagation approaches. Besides, conditional trust propagation model is not discussed carefully in these trust systems. Concretely, with the same local trust and similarity computing mechanism, how different trust propagation models affect the global trust computing even the performance of trust system? What kind of specific factors in trust propagation models are critical and how they affect the performance of propagation process? How robust and resilient the existing trust models are in anticipation of malicious or dishonest feedbacks, and whether the proposed trust propagation model will remain to be effective in the presence of some known attacks? We believe the understanding of these questions is very important for building a more sophisticated reputation management system.

Based on the questions mentioned above, we investigated several popular trust propagation models in reputation management systems. The main contributions of this paper can be summarized as follows. (i) We investigated the two most important trust propagation models in this paper: uniform propagation and conditional propagation, which could be subcategorized into two classes, the similarity based and the local trust value based. (ii) We discuss several important issues affecting the performance of trust propagation models, like how to avoid trust sink/malicious clique, how to design the jump strategies for avoiding trust sink. (iii) We also argue that trust propagation models also should have decay factor to emphasize the most recent history and discount the remote history. (iv) In our study, we found that uniform propagation is hard to resist sophisticated attack behavior and we identify conditional propagation with similarity reference is more attack resilient under the same circumstance. In the similarity based conditional propagation, we identify that the way to use similarity is very important. Should we use similarity only as weight or a threshold value for cutting some path in conditional propagation? Experimental evaluation with independent and colluding attack models shows that only similarity based conditional propagation is attack resilient in most cases but still vulnerable in certain cases. We should combine them together.

The rest of this paper is organized as follows. Section 2 briefly describes the related work. Section 3 discussed the uniform and conditional trust propagation models and formulized them. Besides, Section 3 also gives an example network to show how these different trust propagation models work. Section 4 discussed the threat models we use briefly. We report our experimental evaluation results in Section 5 and conclude the paper in Section 6.

2. Related Work

From the standpoint of trust, a trust network is naturally modeled as a weighted directed graph. Each of the real existed edge in this directed graph corresponds with a trust relationship from the node at the source vertex to the node at the target vertex. The weight on the edge could be treated as a measure of how strong this trust relationship is. The trust propagation process could help us building a path from a source vertex to a target vertex without a direct edge.

Nowadays, existing trust models are categorized into two categories from the standpoint of trust propagation: nonpropagating trust model and propagating trust model. One of the famous nonpropagating trust models is PeerTrust [6]. The feedback credibility concept and its role for defending against dishonest feedback were first introduced in PeerTrust, which also described the roles transaction context may play in making trust model more resilient to attacks. Another trust model without trust propagation process is proposed by [9], for supporting trust in virtual communities by direct experiences and reputation. The semantic distance of feedback ratings was also introduced in this paper. Reference [10] proposed a supervised learning approach that automatically predicts trust between a pair of users using evidence derived from actions of individual users (user factors) as well as from interactions between pairs of users (interaction factors). This method is completely detached from the structure of network graph. References [11–13] also discussed trust evaluation models relying on past observed behaviors, which can also be subcategorized in nonpropagating classification. In recent years, game theory [14–16] was introduced in trust computing, which is another example of trust computation without trust propagation. Most of the work in this area is trying to design some mechanism to make people cooperate with each other rather than defecting. Although there have been some preliminary attempts at studying game theory [17, 18] on trust among agents, game theory is still not among the primary focuses of research in trust management.

Trust models with trust propagation are more popular than nonpropagation trust models. Intuitively, trust propagation could without doubt handle the sparse problem in a trust network. Basically most of the trust propagation models are based on uniform propagation framework, which means users will propagate their trust value to others following all their direct trust relationships or edges in the trust network. One of the most famous trust models with uniform trust propagation is EigenTrust, which actually applied the propagation principle of PageRank [19] in the propagation process with a little bit of changes on the jump strategies and weight of the edges. EigenTrust shows good attack resilience on specific simple threat models. But it could not have a satisfied performance when encountering more sophisticated threat models. ServiceTrust [8] is another trust model applying similarity at the propagation process. ServiceTrust analyzes the vulnerabilities of EigenTrust and shows a better performance on some more sophisticated attack models by introducing feedback similarity weighted trust propagation into the trust model. Reference [20] argued that, in the trust inference process, the trust information and confidence on the trust estimate should be separated and proposed a new trust algorithm to compute an estimate of trust based on only those information sources with the highest confidence estimates. Another interesting concept introduced in trust propagation process is how distrust is propagated [21]. Ortega et al. [22] proposed a novel system intended to propagate both positive and negative opinions of the users through a network, in such a way that the opinions from each user about others influence their global trust score.

3. Trust Propagation Models

In this section, we first briefly introduce the uniform propagation models, including the nonpropagating trust models and propagating trust models. We argue that uniform nonpropagating trust models could be seen as the special situation of trust propagation models with 1-step propagation. We listed some critical factors about feedback applied in trust propagating process and discussed some general ways to get them. In addition, we also discussed other important factors in the trust propagation process, including the strategies of jumping out of malicious clique [16] and the decay factors in the trust propagation. Then we proposed the conditional trust propagation model and introduced two states of each participant in conditional trust propagating process.

3.1. Preparation of Trust Propagation. In reputation management systems, each participant can be a requestor (trustor) and also a responder (trustee). Thus the relationship between any two pairs of participants simulates the peer to peer relationship in any complex network. Hence, we refer to a member of the reputation management system by either "participant" or "peer" in the rest of the paper. Each participant i's reputation could be represented by the global trust value of i. The basic idea of trust propagation is based on "circle of friends" with $k(k \geq 0)$ hops to get the global trust value of each user.

Before we start the trust propagating process, we need to compute the local trust value of each peer by aggregating all the feedback ratings that the peer has received from other participants. Indeed, local trust values provide valuable reference for trust enabled selection of trustor when the local trust network is dense enough. In a feedback based trust propagation models, three important factors are involved for the local trust computation and aggregation as follows.

(1) Rating Mechanism. When a peer P_i acting as a provider to respond to a service request from another peer P_j, the receiving peer P_j is allowed to enter a feedback rating on P_i in terms of the quality of the service provided by P_i. We refer to this as a per-transaction based feedback. Usually, there are two mechanisms that peer gives their feedback. (i) Binary rating mechanism: let $\mathrm{tr}(i, j)$ denote the feedback rating from peer i to peer j where $i, j \in [1, \ldots, n]$ and $\mathrm{tr}(i, j)$ is initialized to zero. With a binary rating scheme, when peer i competes a transaction with another peer j at time t, then peer i may rate the transaction it has with peer j as positive by $\mathrm{tr}(i, j, t) = 1$ or negative by $\mathrm{tr}(i, j, t) = -1$. The key advantage of binary rating is that it could get a number of discrete numbers of 1 or −1, which is easy to be formulated mathematically. (ii) Multiscale rating mechanism: binary rating mechanism is so simple that it loses a lot of useful information and brings some drawbacks which make the reputation system more vulnerable. In the first place, rating mechanism should be more in accordance with human beings' rating habit. Commonly, people would like to judge a transaction in a more subjective way, such as "not bad," "good," and "excellent". In the next place, it is easy for us to distinguish "bad" and "good" from binary mechanism, but it is hard to distinguish different "good" or "completed" services. There is no difference between "an excellent" service and an ordinary "good" service. This mechanism will discourage people to try to make their service better but to muddle along. In addition, the reputation system based on binary mechanism is too simple to be attack resilient for some special threat models. Based on these considerations, multiscale rating mechanism [8] is proposed as a more subjective and more incentive way to encourage users to make their service better and also could punish the malicious ones to make the system more attack resilient. With the same definition of $\mathrm{tr}(i, j)$ as mentioned in binary rating, $\mathrm{tr}(i, j)$ is formulated as follows:

$$\mathrm{tr}(i, j) = \begin{cases} -1 & \text{bad} \\ 0 & \text{no rating} \\ 1 & \text{neutral} \\ 2 & \text{fair} \\ 3 & \text{good} \\ 4 & \text{very good} \\ 5 & \text{excellent.} \end{cases} \quad (1)$$

(2) Feedback Aggregating Mechanism. This computation process could be referred to as the local trust computation in a mathematical formulation, which is the abstract mathematical specification of how the available information should be transformed into a usable metric. Lots of ways are discussed with respected to local trust computation. One of the most intuitive mechanisms is proposed in EigenTrust, which just got the difference between satisfactory transactions and

unsatisfactory transactions. Let $s(i,j)$ denote the total number of satisfactory transactions between peer i and peer j and $\text{unsat}(i,j)$ denote the number of unsatisfactory transactions between peer i and peer j. Hence, $s(i,j) = \text{sat}(i,j) - \text{unsat}(i,j)$. Clearly, $s(i,j)$ is a positive integer if there are more positive feedback ratings and negative otherwise. After analyzing the vulnerabilities of this intuitive way of local trust computing, ServiceTrust [8] proposed a new, more attack resilient local trust computation formula based on the multiscale rating mechanism and user's rating variance. Other proposals include the use of Bayesian procedures [23, 24] or fuzzy decision logic [25] to transform the feedback rating information into ratio-scaled local trust values. Literature [13] described a new system called the beta reputation system which is based on using beta probability density functions to combine feedback and derive reputation ratings.

(3) Normalization Mechanism. After we got the local trust value that participant i places on participant j, $s(i,j)$, it is important to normalize it into the $[0,1]$ interval to make the comparison meaningful between peers with high volume of transactions and peers with low volume of transactions. We can use i's trust on all the rest of players to obtain a normalized $s(i,j)$ for a specific j, say, $c(i,j)$.

In a nonpropagating trust models, we could get the global trust value of peer i by aggregating all the direct trust values from i's direct neighbors; t_i could be represented by the following:

$$t_i = \sum_{j=1}^{N} c(j,i) \quad j \neq i. \tag{2}$$

Clearly, there are many proposals with different discussion of these three factors computation. We need to select appropriate algorithms based on different realistic applications.

3.2. Uniform Propagating Trust Models. Computing the reputation of a peer in a nonpropagating trust model is basically only based on the direct trust relationship between users. But in a large sparse complex networks, a peer i only has very few direct trust values to other peers and, to most of the rest, they are strangers to peer i. Hence we need the *k-hop* propagating process to find a more accurate value to assess i's reputation in the whole system. When $c(i,j) = Null$, we will compute the trust value i places on j, based on the circle of friends that i and j have in common. This circle can be defined by *k-hop* traversal reachability from i to j in a trust graph and $k > 1$. Indeed, the nonpropagating trust models could be referred to as the special case of $k = 1$. When $k = 1$, the global trust value $t(i,j) = \sum_k c_{ik}c_{kj}$. This could be referred to as 1-hop indirect transition probability.

Let n denote the total number of participants in a reputation system. We define $C = [c_{ij}]$ as a matrix of n rows by n columns. Let t_i^0 denote the initial global trust value of peer i. Let t_i^k denote the *k-hop* global trust value of peer i. We have $t_i^1 = \sum_{j=1}^{n} c_{ji}/n = \sum_{j=1}^{n} c_{ji}t_i^0$.

Let $\vec{t}^k = (t_1^k, \ldots, t_i^k, \ldots, t_n^k)$ denote the global trust vector of size n at kth iteration $(1 \le k < n)$. The general formula for computing the global trust vector at the $(k+1)$th iteration is

$$\vec{t}^{k+1} = C^T \vec{t}^k \qquad c_{ij} = \begin{cases} \neq 0 & \text{if } s_{ij} \neq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Based on formula (3), when $k = 0$, we have $\vec{t} = (C^T)^m \vec{t}^0$.

The trust vector $\vec{t}$ will converge to the left principal eigenvector of C if m is large enough [16]. For each element of this global trust vector, t_i, it quantifies how much trust the system as a whole places on the participant i.

This general form of global trust computing suffers several practical issues listed as follows. (i) Cold start problem: when i has not had any transaction with anyone, it has no rating and thus no trust value. Specifically, how we initialize everyone's global trust value when $k = 0$? (ii) Graph clique: by analyzing the basic propagating mechanism in formula (3), we could easily find that if there exist some cliques in the trust network topology, users in the clique/collective would boost each other's global trust value repeatedly. Malicious users could easily take advantage of this feature and mislead the system. In the uniform trust propagation models, we will discuss several typical ways to handle these issues and compare them, respectively. In the following paragraphs, we abbreviate *"Uniform Trust Propagation" as UTP.*

In the following, we will discuss several different uniform trust propagation models.

3.2.1. UTP1: UTP with Uniform Jump Strategy and Initialization. In this way of trust propagation, we have $t_i^0 = 1/n$. Namely, we initialize the global trust values for all participants based on a uniform probability distribution over all n peers. Besides, we could introduce the parameter alpha α to jump out of the trust sink due to malicious clique. In this trust propagation model, each node could jump to any of the users in the trust network with the same probability.

Let $\vec{t}^k = (t_1^k, \ldots, t_i^k, \ldots, t_n^k)$ denote the global trust vector at kth iteration. The general formula for computing the global trust vector at the $(k+1)$th iteration is

$$\vec{t}^{k+1} = (1-\alpha) C^T \vec{t}^k + \alpha t^0, \qquad c_{ij} = \begin{cases} \dfrac{1}{\text{out\_degree}(i)} & \text{if } a_{ij} = 1 \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where

$$A = [a_{ij}], \quad a_{ij} = \begin{cases} 1 & \text{if } i \text{ trust } j \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Indeed, A is the adjacent matrix of trust network. In this case, if there is a directed link from i to j, then i places the same trust value on each of his neighbors. This type of trust propagation is actually how PageRank computes the rank value of each page in the network.

3.2.2. UTP2: UTP with Pre_Trust Jump Strategy and Uniform Initialization. In this ways of trust propagation, we have $t_i^0 = 1/n$, which is the same as UTP1. By utilizing a set P of pretrust seed peers as the bootstrap peers, any participant that does not know whom to trust can always trust one of the pretrust peers with a probability of $1/|P|$. Another role of using pretrust peers to serve as some sort of central authority is to handle malicious collectives, namely, those peers who aim at manipulating the system by giving each other high local trust values and giving all others low local trust values in an attempt to gain high global trust values. A common way to break the collectives is by having each participant placing some trust on pretrust seed peers that are definitely not a part of the collectives. This will avoid getting stuck within a malicious collective.

Let $\vec{t}^k = (t_1^k, \ldots, t_i^k, \ldots, t_n^k)$ denote the global trust vector at kth iteration. The general formula for computing the global trust vector at the $(k+1)$th iteration is

$$\vec{t}^{k+1} = (1-\alpha) C^T \vec{t}^k + \alpha \vec{p},$$
$$c_{ij} = \begin{cases} \dfrac{1}{\text{out\_degree}(i)} & \text{if } a_{ij} = 1 \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where

$$p_j = \begin{cases} \dfrac{1}{|P|} & \text{if } j \in P \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

3.2.3. UTP3: UTP with Pre_Trust Jump Strategy and Pre_Trust Initialization. Compared with UTP2, the only difference in UTP3 is the way we assign initial global trust values. We have $t_i^0 = \vec{p}$.

The common point of these three trust propagation models is no matter how user *i's* neighbors perform, they are equal with respect to local trust values for peer *i*. In most of the reputation systems, this kind of trust models usually is not attack resilient. We could see that with a pre_trust jump strategy, the malicious users are hard to make a graph clique to boost each other's global trust value. So, based on these analyses, we could get the result that, in the same context, the attack resilience of UTP3 is much stronger that UTP2 and UTP2 are much stranger than UTP1. We will give an example experiment to show that.

3.2.4. UTP4: UTP3 with Local Trust Weight Propagation. In this way of trust propagation, we have $t_i^0 = \vec{p}$. Compared with UTP3, instead of assigning the same local trust value to each of his neighbors, peer *i* will propagate his trust value based on the feedback ratings.

Let $\vec{t}^k = (t_1^k, \ldots, t_i^k, \ldots, t_n^k)$ denote the global trust vector at kth iteration $(k > 1)$. Based on this propagation mechanism, the general formula for computing the global trust vector at the $(k+1)$th iteration is

$$\vec{t}^{k+1} = (1-\alpha) C^T \vec{t}^k + \alpha \vec{p},$$
$$c_{ij} = \begin{cases} \dfrac{\max(s(i,j),0)}{\sum_j \max(s(i,j),0)} & \text{if } \sum_j \max(s(i,j),0) \neq 0 \\ p_j & \text{otherwise,} \end{cases} \tag{8}$$

where

$$p_j = \begin{cases} \dfrac{1}{|P|} & \text{if } j \in P \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The computation of $s(i,j)$ is discussed in [5]. And since what we focus on is the propagation mechanism, we would not discuss it in detail.

3.2.5. UTP5: UTP4 with top_q Jump Strategy and Pre_Trust Initialization. Considering the jumping choice of a peer *i*, we find that it only has two selections: everyone or pre_trust ones. Here we proposed another jump strategy, called *top_q* jump strategy. Instead of jumping to the pre_trust users in the trust propagation, a user would like to choose a set of Q users whose global trust values are in the *top_q* list. Namely, a user always tries to believe the user who has a higher global trust value. This way of trust propagation could be formulated as follows.

In this way of trust propagation, we have $t_i^0 = \vec{p}$. Compared with UTP3, instead of assigning the same local trust value to each of his neighbors, peer *i* will propagate his trust value based on the feedback ratings.

Let $\vec{t}^k = (t_1^k, \ldots, t_i^k, \ldots, t_n^k)$ denote the global trust vector at kth iteration $(k > 1)$. Based on this propagation mechanism, the general formula for computing the global trust vector at the $(k+1)$th iteration is

$$\vec{t}^{k+1} = (1-\alpha) C^T \vec{t}^k + \alpha \vec{q},$$
$$c_{ij} = \begin{cases} \dfrac{\max(s(i,j),0)}{\sum_j \max(s(i,j),0)} & \text{if } \sum_j \max(s(i,j),0) \neq 0 \\ p_j & \text{otherwise,} \end{cases} \tag{10}$$

where

$$q_j = \begin{cases} \dfrac{1}{|Q|} & \text{if } j \in Q \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Intuitively speaking, *top_q* jump strategy looks more reasonable in a real life and more decentralized since it does not depend on the pre_trust users anymore. However, once the malicious users are included in the *top_q* list, the consequence is fatal. We will show the experiment result of this kind of jump strategy in Section 3.4. This explains that why we select pretrust users to jump rather than others.

It is also worth mentioning that the *top_q* jump strategy may cause the trust propagation process unconverging. Once different malicious users come into the *top_q* users in different iterations, it may make the malicious users' global trust values fluctuate sharply in different iteration rounds.

3.2.6. UTP6: UTP4 with Decay Factor d. Another design consideration is that we need to differentiate recent trust history from remote past history. So we need to introduce a "forgetting factor" to emphasize this consideration, namely, the decay factor d. Considering one important attack strategy of malicious users-malicious clique, we believe that decay factor could effectively improve the attack resilience of trust models.

Concretely, let $d \in (0,1]$ be the decay factor. By incorporating the decay factor in the trust propagation formula (8), we can compute the global trust at the $(k+1)$th iteration by formula (12) as follows:

$$\vec{t}^{k+1} = d \times (1-\alpha) C^T \vec{t}^k + \alpha \vec{p},$$
$$c_{ij} = \begin{cases} \dfrac{\max(s(i,j),0)}{\sum_j \max(s(i,j),0)} & \text{if } \sum\limits_j \max(s(i,j),0) \neq 0 \\ p_j & \text{otherwise,} \end{cases} \tag{12}$$

where

$$p_j = \begin{cases} \dfrac{1}{|P|} & \text{if } j \in P \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

3.3. Conditional Trust Propagation Models. The basic idea of uniform trust propagation is that if there is a directed link from participant i to participant j in the trust network, i would always trust certain amount of trust value to j no matter how weak this trust relationship is. But in a realistic trust system, if the trust relationship between two participants is too weak, i would like to cut this relationship and does not propagate its trust value to j, or j is in an inactive status with respect to participant i. If the trust relationship is higher than the threshold value, we say j is active with respect to peer i. We call this kind of trust propagation threshold value based trust propagation. Figure 1 gives an example of conditional trust propagation and uniform propagation.

Figure 1(a) shows the initialized trust network and the value on the edge could be the local trust value that a trustor placed on a trustee or some other type of values, for example, the feedback similarity value. Figure 1(b) depicts the uniform trust propagation; we could see that even user id = 2 places very low weight value on user id = 3; it still propagates its trust to it. But in Figure 1(c), that way is cut since the weight value is less than the threshold value. By this way, suppose that id = 3 and id = 4 are malicious users; they could not get higher profit by the trust propagation process. This will make our trust model more attack resilient. The conditional trust propagation is described in formula (14).

Let $t_i^0 = \vec{p} = \{1/p, \cdots 1/p, 0, \ldots, 0\}$ be the initialization global trust vector. $\vec{t}^k = (t_1^k, \ldots, t_i^k, \ldots, t_n^k)$ denotes the global trust vector of size n network at kth iteration. The general formula for computing the global trust vector at the $(k+1)$th iteration is

$$\vec{t}^{k+1} = (1-\alpha) \times \text{th} \times C^T \vec{t}^k + \alpha \vec{p},$$
$$\text{th} = \begin{cases} 1 & \text{if } c_{ij} > \text{threshold value} \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

where

$$p_j = \begin{cases} \dfrac{1}{|P|} & \text{if } j \in P \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Formula (14) is the local trust based threshold value conditional trust propagation. We call it *LTCP*.

3.4. Similarity Based Trust Propagation: Uniform and Conditional. Similarity based technologies have been widely used in reputation management systems [26–28]. Similarity based approaches can be categorized in two groups: feedback based similarity and profile based similarity. There are a lot of methods to compute the pairwise similarity. One of the most common ones is based on Euler distance between two users' rating behavior vectors. Another popular way to computer similarity is cosine function based which computes the cosine value between two rating vectors. How to compute the similarity is another important area in trust computation and collaborative filtering. In this paper, what we focus on is how to introduce similarity value into our trust propagation process and what kind of influence it will lead. Based on this discussion, we proposed our similarity-related trust propagation process in both uniform propagation and conditional propagation.

Let $\text{Sim} = [\text{sim}_{ij}]$ $0 \leq \text{sim}_{ij} \leq 1$ indicate the similarity matrix of the trust network. sim_{ij} is the similarity, either feedback behavior similarity or profile similarity, between participant i and participant j.

3.4.1. SLUTP: Similarity and Local Trust Weighted Uniform Trust Propagation. When peer i propagates his trust value to its neighbors, it will consider the local trust that it places on each neighbor as the propagation weight on the first layer. Then, by computing the similarity between i and its neighbor, the similarity value is placed as the propagation weight on the second layer. This way of trust propagation is proposed in ServiceTrust.

Let $t_i^0 = \vec{p} = \{1/p, \cdots 1/p, 0, \ldots, 0\}$ be the initialization global trust vector. $\vec{t}^k = (t_1^k, \ldots, t_i^k, \ldots, t_n^k)$ denotes the global trust vector of size n network at kth iteration. The general formula for computing the global trust vector at the $(k+1)$th iteration is

$$\vec{t}^{k+1} = (1-\alpha) \times \text{Sim}^T \times C^T \vec{t}^k + \alpha \vec{p},$$
$$c_{ij} = \begin{cases} \dfrac{\max(s(i,j),0)}{\sum_j \max(s(i,j),0)} & \text{if } \sum\limits_j \max(s(i,j),0) \neq 0 \\ p_j & \text{otherwise,} \end{cases} \tag{16}$$

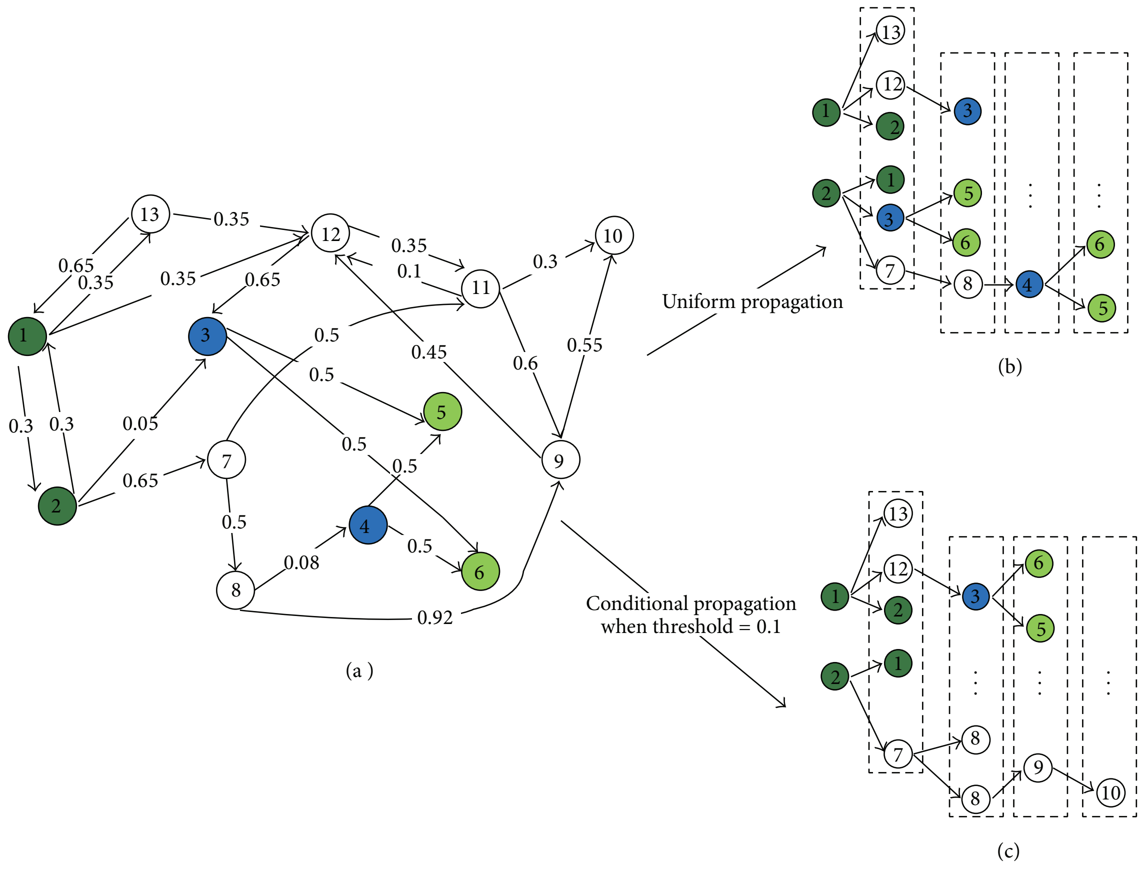


Figure 1: Comparison of uniform trust propagation and conditional propagation.

where

$$p_j = \begin{cases} \frac{1}{|P|} & \text{if } j \in P \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

Similarly, we could introduce the decay factor into *SLUTP*, which is called *D-SLUTP*. Formula (16) could be rewritten as follows:

$$\vec{t}^{k+1} = d \times (1-\alpha) \times \mathrm{Sim}^T \times C^T \vec{t}^k + \alpha \vec{p},$$
$$c_{ij} = \begin{cases} \frac{\max(s(i,j),0)}{\sum_j \max(s(i,j),0)} & \text{if } \sum_j \max(s(i,j),0) \neq 0 \\ p_j & \text{otherwise,} \end{cases} \tag{18}$$

where

$$p_j = \begin{cases} \frac{1}{|P|} & \text{if } j \in P \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

In formula (18), if we take the $C = I$ (I is unit matrix) and $d = 1$, we got the similarity weighted uniform propagation. In this situation, peer i will propagate its trust values based on its rating behavior similarity with respect to other neighbors. If we take the $\mathrm{Sim} = I$ and $d = 1$, then we get the local trust weighted uniform propagation.

3.4.2. STCP: Similarity Threshold Value Based Conditional Trust Propagation. We already discussed the general form of conditional trust propagation in Section 3.2. In STCP, we introduce the pairwise similarity as the condition that i with respect to j is active or not. Namely, if the feedback based similarity between i and j is higher than the predefined threshold value, then i is activated with respect to j and has the willing to propagate its trust value. STCP could be formulized in (20).

Let $t_i^0 = \vec{p} = \{1/p, \cdots 1/p, 0, \ldots, 0\}$ be the initialization global trust vector. $\vec{t}^k = (t_1^k, \ldots, t_i^k, \ldots, t_n^k)$ denotes the global trust vector of size n network at kth iteration and d is the decay factor. The general formula for computing the global trust vector at the $(k+1)$th iteration is

$$\vec{t}^{k+1} = d \times (1-\alpha) \times \mathrm{th} \times C^T \vec{t}^k + \alpha \vec{p},$$
$$\mathrm{th} = \begin{cases} 1 & \text{if } \mathrm{sim}_{ij} > \text{threshold value} \\ 0 & \text{otherwise,} \end{cases} \tag{20}$$

where

$$p_j = \begin{cases} \frac{1}{|P|} & \text{if } j \in P \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

In some type of attacks, malicious users would like to give some honest ratings or profiles with a probability to gain some similarity with good ones. This similarity based conditional trust propagation could effectively prevent this kind of camouflage behavior.

3.4.3. HCP: Hybrid Conditional Trust Propagation. In some cases, we could not cut all the malicious users out of the trust propagation only depending on the similarity based threshold value. For example, if our threshold value is 0.5 and the spy users' similarity value with respect to other good ones is 0.51, we cannot cut this one out. What we need to do is to make sure that even if this spy user comes into the trust propagation process, we still could minimize its negative influence on the reputation management system. In order to improve the robust of trust model, we proposed this new hybrid conditional trust propagation process. The main idea of HCP could be divided into two steps before participant i decides to propagate its trust value to others. First, it will check its similarity value with his neighbors in the trust network to find if it is higher than the threshold value or not. Then if the similarity value is higher enough, user i still uses the similarity as the weight in the trust propagation. HCP could be formulized as follows.

Let $t_i^0 = \vec{p} = \{1/p, \cdots 1/p, 0, \ldots, 0\}$ be the initialization global trust vector. $\vec{t}^k = (t_1^k, \ldots, t_i^k, \ldots, t_n^k)$ denotes the global trust vector of size n network at kth iteration and d is the decay factor. The general formula for computing the global trust vector at the $(k+1)$th iteration is

$$\vec{t}^{k+1} = d \times (1-\alpha) \times \text{th} \times \text{Sim}^T C^T \vec{t}^k + \alpha \vec{p},$$
$$\text{th} = \begin{cases} 1 & \text{if } \text{sim}_{ij} > \text{threshold value} \\ 0 & \text{otherwise,} \end{cases} \tag{22}$$

where

$$p_j = \begin{cases} \dfrac{1}{|P|} & \text{if } j \in P \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

When the threshold value is 0 and the decay factor is 1, the propagation model described in formula (22) becomes into the SLUTP. If the threshold value is 1 and the similarity matrix Sim $= I$, we get the UTP6 or UTP4 depending on the decay factors value. Therefore, by exploring different boundary-values of these factors in formula (22), we could get all kinds of trust propagation models we discussed above.

3.5. Illustration of Comparison among These Trust Propagation Models. In this section, we will compare these trust propagation models based on the same trust network which is depicted in Figure 2. In this network, there are four types of participants: pretrust participants, normal participants, spy participants, and normal malicious participants.

There are some basic rules when we compute the local trust values and pairwise similarity values. For the local trust value, each user's local trust values placed on its trust neighbors follow the *zipf* distribution (*weighted propagation*) or *uniform* distribution (*unweighted propagation*) and the sum should be 1. For the similarity value, the similarity between spy users and good ones (including pretrust and normal) should be less than the similarity between good ones and good ones. We suppose there are no similarity between good users and normal malicious users. The spy users could propagate their trust values to normal malicious users. The attack model implicated in this trust network has been discussed in [5, 8].

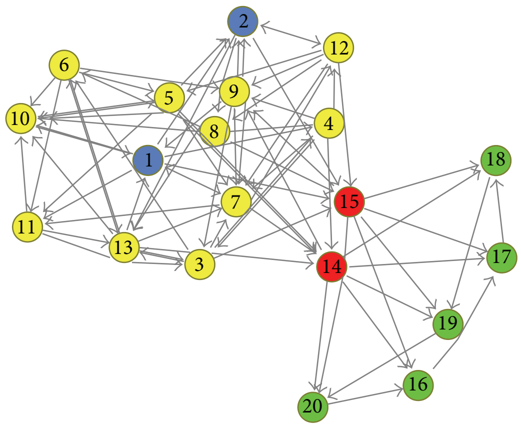


Figure 2: Example service provision trust network with attack behaviors.

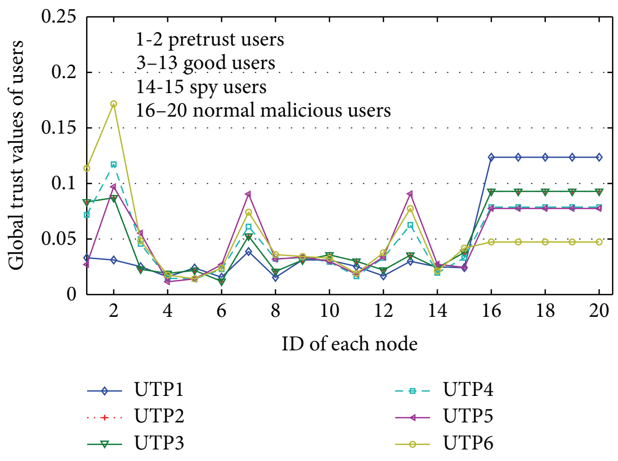


Figure 3: Performance comparison on uniform propagation (d = 0.5 *in UTP6*).

In the first comparison (Figure 3), UTP6 deserves the best performance in which the malicious users could not obtain high global trust values but good ones are trusted more by the system after propagating process converging. The worst case is UTP1 in which the malicious users are trusted more by the system. The global trust values of malicious users are way higher than the good ones. This will make the malicious users subvert the system easily. By comparing UTP1, UTP2, and UTP3, we can see that the choice of the initialization vector does not influence the global trust values but the choice of jump strategies does. If we choose the random jump strategy, we can see that malicious users (id 16–20) obtain much higher global trust values. By comparing UTP4 and UTP5, we could

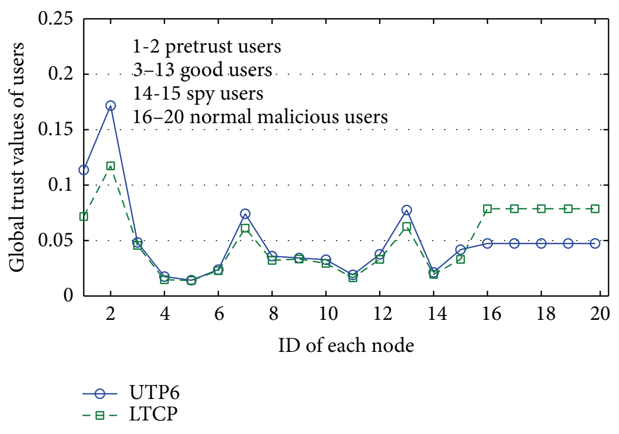


Figure 4: Performance comparison on uniform propagation (UTP6, $d = 0.5$) and conditional propagation (LTCP).

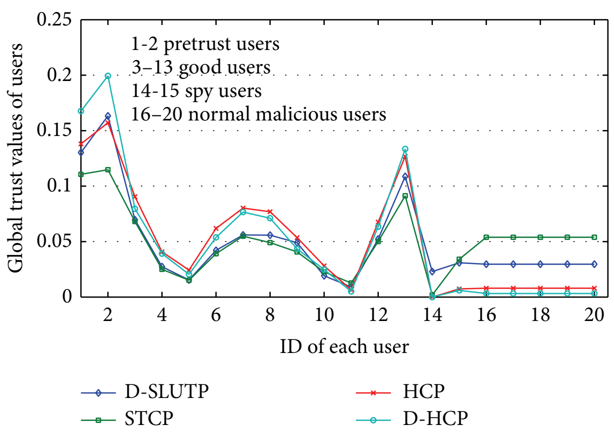


Figure 6: Performance comparison on uniform trust propagation and conditional trust propagation ($d = 0.5$).

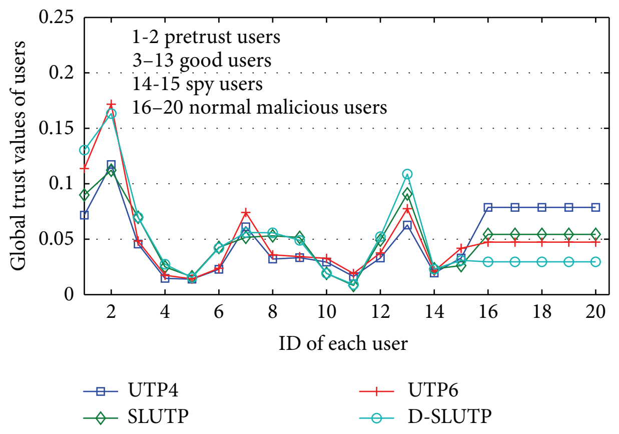


Figure 5: Performance on uniform trust propagation (UTP4, UTP6, SLUTP, and D_SLUTP) $d = 0.5$.

see that UTP4 is a little bit better than UTP5. But both of them are worse than the UTP6. By carefully observing the trust propagating process in UTP5, the spy peer id = 15 is selected in the *top_q* ($q = 3$) users which makes the other normal malicious users boosted by spy user id = 15.

Figure 4 shows the performance comparison of UTP6 and LTCP. Clearly, uniform trust propagation with decay factors could effectively hinder normal malicious users obtaining high global trust values by boosting each other in the malicious collection. Due to the fact that spy users also always provide good services or files when responding to a request, if we take the local trust as our threshold value reference, we could not effectively prevent the trust propagated from good peers to spy users.

In Figure 5, we could see that, by introducing similarity into the trust propagation process, the attack resilience of trust propagation process is effectively improved. The malicious users in SLUTP only could obtain a little bit higher global trust values than the UTP6 which includes the decay factor but significantly lower global trust values than the UTP4. If we introduce both decay factor and similarity into the uniform trust propagating process, the attack resilience of trust system is obviously improved. The global trust values of malicious users in D_SLUTP are lowest in Figure 5 and most of good ones are higher than other propagating models.

Since the D_SLUTP shows the best performance among the four trust propagating trust models in Figure 5, we take D_SLUTP as our benchmark in the following comparison. Figure 6 shows the attack resilient performance of D_SLUTP and other similarity included conditional trust propagation models. The D_SLUTP shows the worst performance among these four trust propagating models. The normal malicious users could hardly get any global trust values in HCP and D_HCP by carefully utilizing similarity between spy users and good ones as our threshold value reference and propagating weight. The STCP also shows a better performance than D_SLUTP but worse than the HCP and D_HCP. This phenomenon tells us that, instead of only using similarity as propagating weight or threshold value reference, it is a better way to combine them together.

Another important conclusion obtained by these simulating results is that decay factor is very important to prevent malicious behavior in a trust graph with malicious clique.

In order to give a vivid illustration of how trust propagation model could handle the sparsity problem in trust network. We show another set of comparison with two networks: the local trust network, which is without trust propagation (Figure 7) and the 1-hop propagation trust network (Figure 8). We can see that Figure 8 gives us a more dense trust network compared with Figure 7, which shows how trust propagation process addresses the problem of sparsity.

4. Attack Models

In decentralized networks, lots of threat models are used to characterize malicious behaviors of different forms. Reference [8] discussed four threat models in 2013. These four threat models are independently malicious (threat model A), malicious collectives (threat model B), malicious collectives with camouflage (threat model C), and malicious spies (threat model D). These four attack models are gradually more sophisticated than the previous ones. In this paper, we

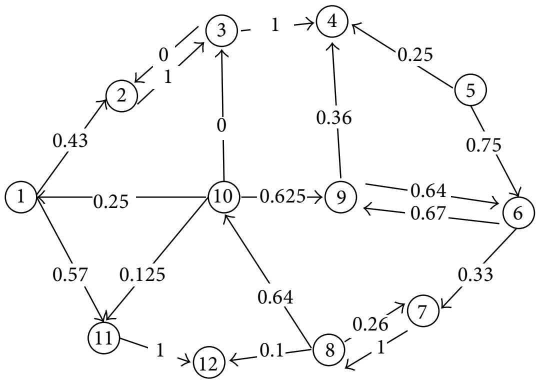


Figure 7: Trust network with normalized local trust values.

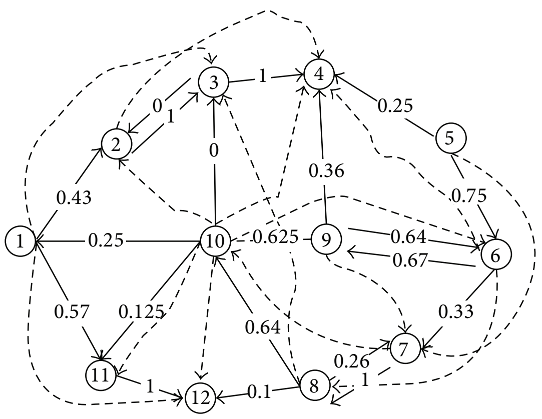


Figure 8: Trust Network after 1-hop trust propagation. Dash line represents the indirect trust computed though 1-hop trust propagation.

proposed another more sophisticated attack model which is malicious spies with camouflage (threat model E).

Threat model E. Malicious participants are also strategically organized into two groups (type E and type B). Compared with type D peers in threat model D, type E peers also provide honest feedback to good peers at $f\%$ when selected as a service provider. In addition, type B peers not only provide bad (inauthentic) services but also form a chain of malicious collective as described in threat model B.

In the following experiments, we will give the evaluation of different kinds of trust propagation models under these five threat models and varying different decay factors and threshold values.

5. Experimental Evaluation

In this section we evaluate the performance of the trust propagation models in terms of attack resilience. Since we already discussed the differences between uniform trust propagation models, we only take UTP1, UTP2, UTP4, UTP6, LTCP, SLUTP, and D_SLUTP as our uniform trust propagation models and STCP, HCP, and D_HCP as our conditional trust propagation models in the Epinion network under five attack models. We also evaluate the effectiveness of uniform and conditional trust propagation models varying the decaying factors. We take the Epinion trust network for our simulating network for a number of reasons. First, Epinion is a platform for people to share their experiences, both good and bad, about a variety of topics, ranging from daily life consumptions (such as cars and coffees) to media objects (such as music and movies). Users could write reviews, rate the reviews of other authors, and, most importantly for our purposes, indicate trust or distrust for another user. Second, the Epinion dataset is a very sparse network. There are about 75879 nodes in Epinion and the degree distribution follows the power-law distribution [26]. Most of the nodes only have limited trust links with other nodes. Basically speaking, it is an ideal sparse network for evaluating the effectiveness of trust propagating process. Figure 9 shows the in-degree and out-degree distribution of Epinion network. In our real experimental setting, due to the limitation of our computation, we only take the first 1000 nodes to run our experiments. The configuration of parameters in the experiments is shown in Table 1.

5.1. Experiment Setup. In the first 1000 nodes, there are 134 nodes without any in-degree which means they are not trusted by any of the 1000 nodes. Three nodes have no out-degree which means they do not trust anybody in the system. The maximum out-degree of 1000 network is 478 and the maximum in-degree of 1000 network is 366. The minimum out-degree and in-degree of the network are both 0. The average degree of the network is 40.24. The degree of this smaller Epinion network still obeys the power-law distribution and is a sparse network. In this 1000-node network, one real preexisting directed link from one node i to another node j represents a direct trust relationship, i trust j in some degree. In our experiments, we take the 134 nodes as malicious node for launching the attacks, including attempting to degrade or destroy the quality and performance of the service network system. And the first 30 users as our pre_trust users whose initial global trust is greater than zero. All the rest are normal good users which are in the network for requiring and responding requests.

In order to keep this power-law distribution network of Epinion, we try our best to not generate new trust links (directed edges) in the existing topology. In this 1000-node Epinion network, we simulate the transactions by query-answering mechanism. Every node could send requests (as service consumer or trustee) and could get a responding list including all its out-degree neighbors. In order to introduce the attack behavior in the transactions, we suppose all the 134 malicious users could answer any of the requests. Only when peer i has received a service provided by peer j, peer i is allowed to give j one feedback.

The simulation of the service network dynamics is done through simulation cycles. Each cycle consists of multiple query cycles. In each query cycle peers can probabilistically choose to ask queries or respond to queries. The number of queries that a peer issued follows the uniform distribution. After issuing a query, peer waits for response. Upon obtaining a list of providers that responded to the query, the peer

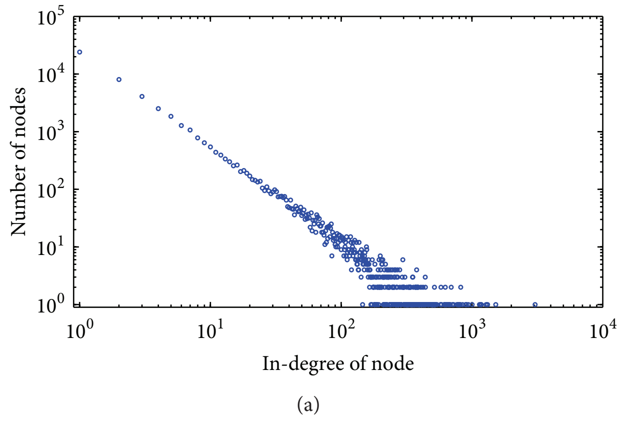


FIGURE 9: In and out degree distribution of Epinion dataset.

TABLE 1: Simulation configuration.

Network scale	1000
Number of malicious nodes	13.4%
Number of pretrust nodes	3%
% of spy users	3.4%
% of service requests in which good user i provides unsatisfied services	0%–10%
% of service requests in which malicious user i provides unsatisfied service	Varied in threat models
% of spy users gives honest ratings	10%, 30%, 50%, 70%
Decay factors	0.1, 0.3, 0.5, 0.7
Threshold value of conditional propagation	0.005, 0.5
Rating mechanism	Multiscale rating mechanism
Similarity algorithm	[8] [Su, et al. 2013]
Local trust algorithm	[5] [Kamvar, et al. 2003]
Initialization strategy	Varied in propagation models
Jump strategy	Varied in propagation models
Queries issued by each node	Uniform distribution 50–100.
Number of cycles	30

selects one provider from the list. The selection process will be repeated until a user has received a satisfying service. We choose the probabilistic-based selection [8] as the selection method. In the conditional trust propagation, we select 0.495 as our similarity referred threshold value. How to properly select a threshold value is also vital in the conditional trust propagation. It is not the key concern of this paper. We will discuss it in detail in our future work.

At the end of each simulation cycle, the local and global trust values are computed. We run each experiment several times and the results of all runs are averaged. The performance metrics used in this paper include the percentage of unsatisfactory services versus the percentage of satisfactory services. If the global trust values accurately reflect each peer's actual behavior, then high global trust values minimize the number of inauthentic downloads. We are also interested in the time complexity and iteration rounds used for trust propagation.

5.2. Evaluation of Trust Propagating Models. In this part, we will show the performance of typical uniform trust propagation models on the Epinion dataset we obtained, including UTP1, UTP2, UTP4, UTP6, SLUTP, and DSLUTP. For the DSLUTP, we take the decay factor $d = 0.5$. All the simulations are executed under threat models A, B, C, D, and E. The experiment results under threat models A and B are depicted in Figure 10.

For threat models A and B, we see a very interesting phenomenon here is that, except UTP1, all the rest uniform trust propagation models show a very good attack resilience, about 20% failed services. UPT1, which could be considered as the original PageRank propagation model, could not handle the least sophisticated threat models A and B. We can see that, by carefully selecting pretrust users in jump strategies, even UTP2 also initializes each peer with a positive global trust values; it still outperforms UTP1 significantly. By comparing UTP2 with UTP4, UTP6, SLUTP, DSLUTP, and conditional trust propagation models, we could see that UTP2 performs a little worse than all the rest models, especially in threat model B (about 23%–25%). The reason is that, with the positive global trust values obtained in the initialization and malicious collective in threat model B, all the malicious users in UTP2 never go to 0.

Another phenomenon is that we could see that, except UTP1 and UTP2, all the rest propagation models are about 20%. By carefully analyzing the global trust values and the topology of trust network, we see that even all the malicious users' global trust values are much lower in these trust propagation models, but constrained by the network topology, there are several good users which have only one other good neighbor to respond to their requests. Once the good neighbor involuntarily provides unsatisfied services, the good users will keep trying all the other responders even they are all malicious users due to our experimental settings.

Firstly we observe the performance of all the uniform trust propagation models on threat model C in Figure 11(a).

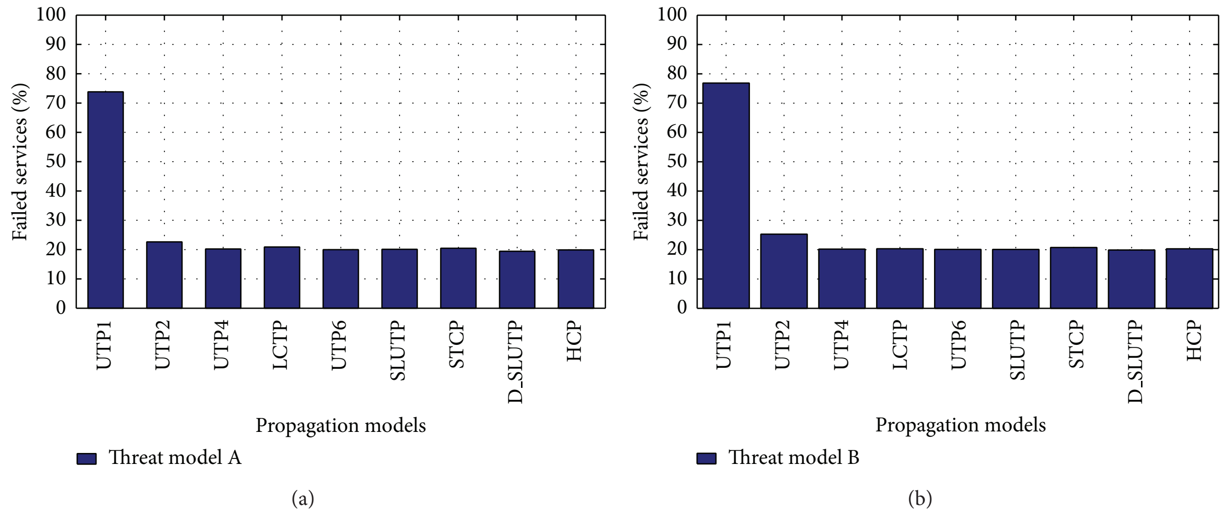


Figure 10: Performance comparison on all trust propagation models under threat models A and B.

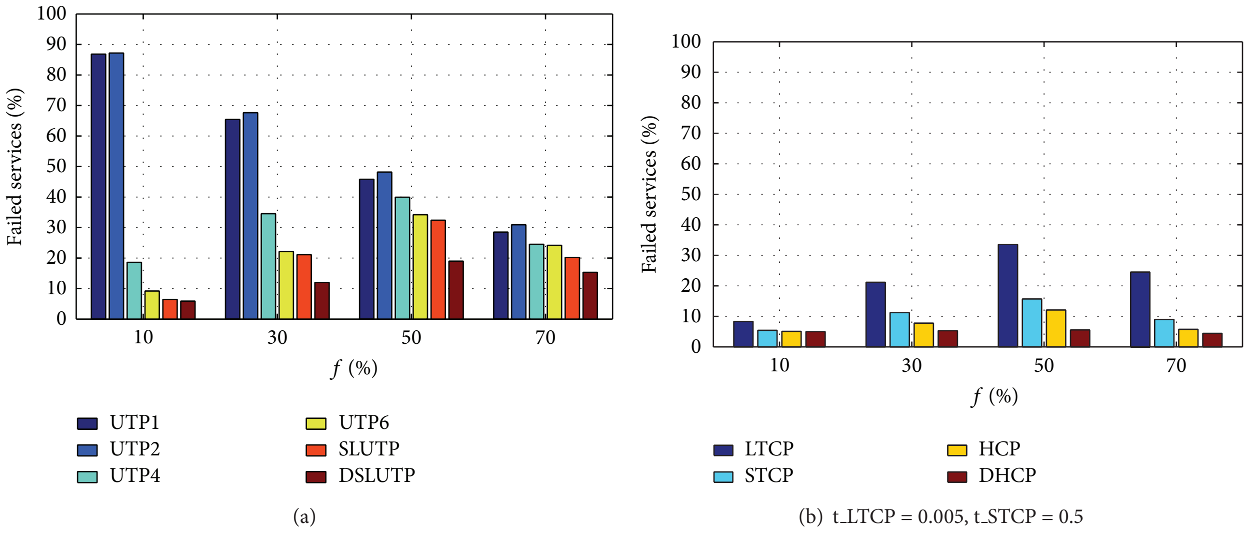


Figure 11: Performance comparison on all trust propagation models under threat model C.

We can see that the DSLUTP outperforms the other uniform trust propagation models in an obvious way. Particularly when the camouflage probability is 50%, there are more than 30% lower compared to the worst case UTP2. By comparing UTP6 with UTP4, DSLUTP with SLUTP we could see that when the malicious users get smarter and try to mislead good users, they do could obtain some positive global trust values from good ones and enhancing their global trust values by malicious clique. However, the decay factor could effectively decrease this kind of bad influence by malicious clique. When the $f\% = 30\%$, UTP6's percentage of failed services is 15% lower than UTP4. And when the $f\% = 50\%$, DSLUTP outperforms SLUTP with about 12% failed services. Figure 11(a) also shows that, by introducing similarity in threat model C, malicious users could not easily get high global trust values any more.

Another very interesting phenomenon in Figure 11(a) that really interests us is that UTP1 outperforms UTP2 slightly in threat model C, which is unexpected. By carefully checking the global trust values in each iteration round, we found that, in UTP2, the pretrust users could get way higher global trust values than they are in UTP1. Since there is no mechanism for preventing the global trust values propagated from pretrust users to malicious users, once the malicious users provide good services to pretrust users, they could get higher global trust values than in UTP1. Due to this reason, the global trust values of malicious users in UTP2 are higher than in UTP1, which makes UTP2 worse (about 3% higher in UTP2).

Figure 11(b) shows the performance of conditional trust propagation models under threat model C. The threshold value for LTCP is t_LTCP = 0.005 (normalized local trust values) and for STCP is s_STCP = 0.5. In this set of experiments we know that even the worst case of conditional trust propagation (LTCP) is all most as good as the SLUTP in the same environment. The reason that LTCP is worse than the DSLUTP case is that we could not cut all the propagation ways from good ones to malicious ones. If we select a higher threshold value in LTCP, it also will cut too more propagation ways that are from good ones to good ones. This will make the trust network too sparse. When the camouflage percentage is 10%, the percentage of failed services of LTCP propagation is less than 10%, which means the camouflage probability is too low to make the malicious users get a higher enough local trust values than the threshold value to make the malicious collective work. But, in other cases, the LTCP performs much worse than the cases that we introduce similarity into the conditional propagation. By carefully applying similarity algorithms and values in conditional trust propagations (STCP), we could improve the attack resilience of trust propagation process obviously (about 11%–20% better than the LTCP). Also, we could see that the HCP and DHCP are even better than the STCP if we use similarity not only as propagation weight but also as reference of threshold value. Particularly when the camouflage probability is 50%, the DHCP is almost 10% better than STCP and 7% better than the HCP, which proves that decay factor could also decline the negative effect of malicious collective.

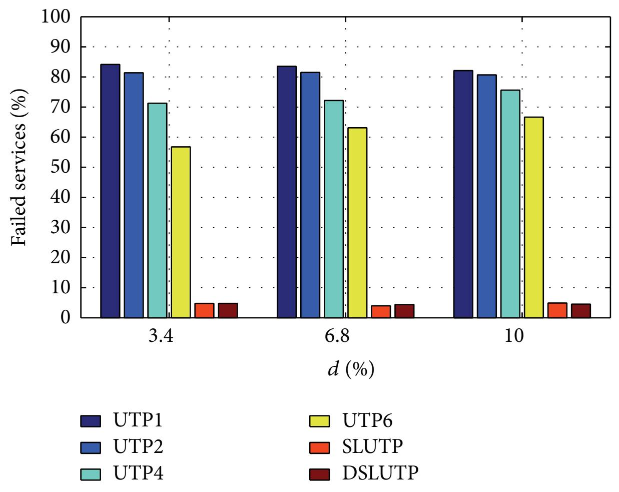
Figure 12 shows each trust propagation models' attack resilience under threat model D, in which we introduce naive spy users and malicious collectives. The first interesting phenomenon is that all the trust propagation models without similarity lost their effectiveness under threat model D (about 57% in UTP 6 is the best result) but all the similarity based trust propagation models show an impressive attack resilience no matter the percentage of spy users out of all the malicious users, which is all at about 5% failed services. By analyzing our similarity algorithm and the malicious behavior in threat model D, we found that the only way that type B users obtain global trust values is from the type D users. However, our similarity algorithm makes it even harder for the spy users to get a positive similarity value between them and good ones. Actually, all the similarity values between spy users and good ones are 0. In this case, there is no trust propagated from good ones to spy nodes and the malicious type-B users could not get any trust either. But without similarity, because spy users perform just as well as good users when providing services; they could not be discovered by good ones only with local trust values. Meanwhile, the phenomenon that UTP6 performs much better than other no-similarity propagation models also proves the decay factor works on some extent.

Figure 13 shows the performance of different trust propagation models under threat model E. In Figure 13(a), we can see that UTP1, UTP2, and UTP4 are not attack resilient at all. The percentage of failed services is all about 80%, which means the global trust values of malicious type B users are much higher than others. Besides, when the $h\%$ is higher than 30%, even the UTP6 and SLUTP and DSLUTP perform much better than other uniform trust propagation models, they still lost their effectiveness on attack resilience. The best case is DSLUTP which is with almost 42%–53% of failed services. This shows that if spy users could obtain some similarity, the uniform trust propagation models fail to defend the trust management against the colluding behavior of spy users and type-B users. But when the probability of honest rating is not high enough, like 10%, the spy users could not get high trust values from good ones because of the too much low similarity values. This makes the 10% of honest rating case with similarity in uniform models show a better performance.

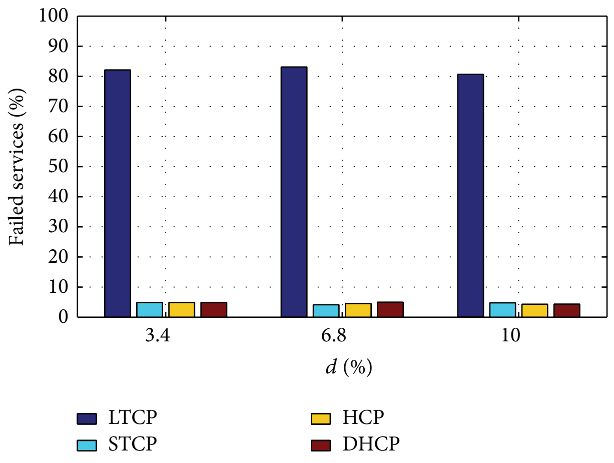
In Figure 13(b), we can see that the percentage of failed services in LTCP is about 84% percentage, which is even about 3% higher than UTP1, UTP2, and UTP4. The LTCP is the worst propagation model in these trust propagation models. That means in this case that the LTCP could not defend the trust management system from malicious users but hurt the propagation between good ones. However, if we introduce the similarity into the conditional trust propagation, we could see it making the trust propagation models more attack resilient. The percentage of failed services in STCP is about 5% higher than the HCP and DHCP when the honest rating probability is 30%, 50%, and 70%. Also, the DHCP shows the best attack resilience among all the trust propagation models. Similarly, with the 10% of honest rating, the similarities between spy users and good ones are too small to make a big difference between STCP and HCP, DHCP. They are all good enough.

5.3. Impact of Different Decay Factors and Different Threshold Values. In this set of experiments, we will take the most sophisticated threat model E with the honest rating $h\%$ = 50% to show how different decay factors and threshold values affect the performance of the trust propagation models. To make a clear comparison, when we explore the threshold value's affection, we take the STCP as our baseline propagation model, and DSLUTP as the baseline propagation model for exploring the affection of decay factors.

By measuring the percentage of failed services with varying settings of threshold values from 0.2 to 0.5 in Figure 14(a), we can see that if the threshold value is too low, such as 0.2 and 0.3, the STCP could not show too much attack resilience on threat model E, but when the threshold value is higher than 0.4, STCP shows a much better performance on attack resilience. By observing the similarity values during the simulation, we could see that most of the similarity values between spy users and good ones are above 0.3–0.5. It is natural to think that the higher the threshold value is, the better of the performance is. However, if we choose a too much higher threshold value, there are also too many propagation ways from good ones to good ones being cut. Actually, as we mentioned before, how to choose a propitiate threshold value could also be an interesting future exploration, such as how they are affected by the algorithm, how they are computed, dynamic threshold value,

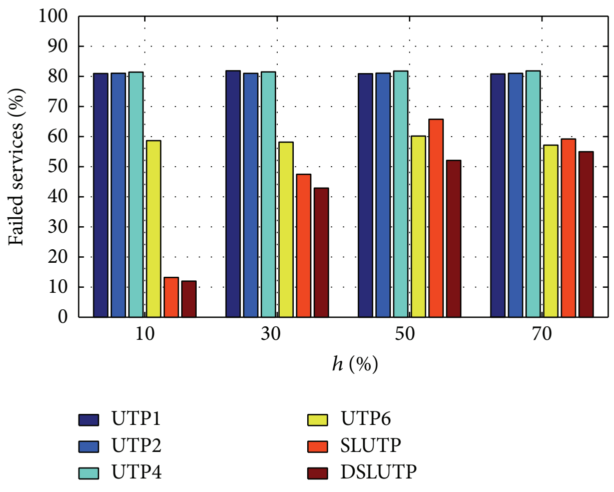


(a)

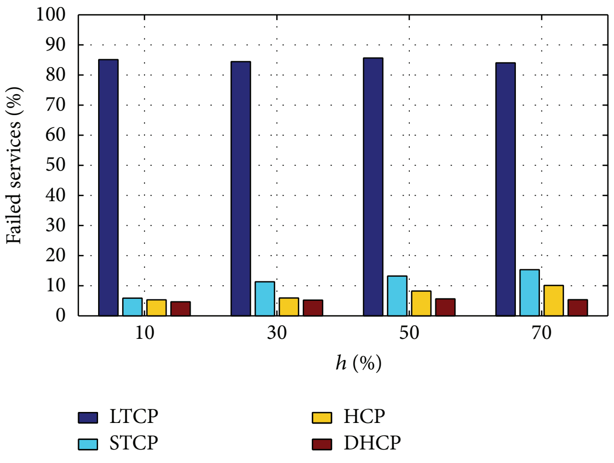


(b) t_LTCP = 0.005, t_STCP = 0.5

FIGURE 12: Performance comparison on all trust propagation models under threat model D.

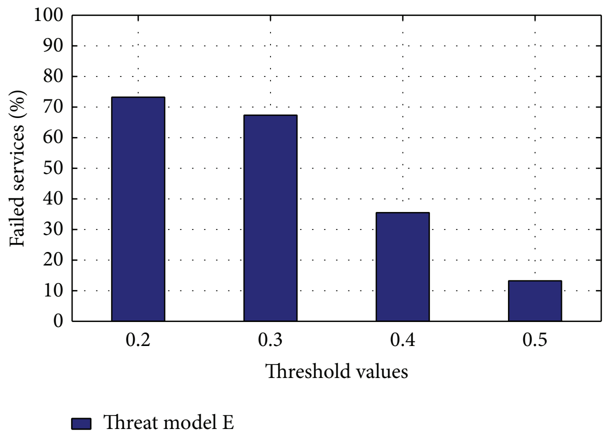


(a)

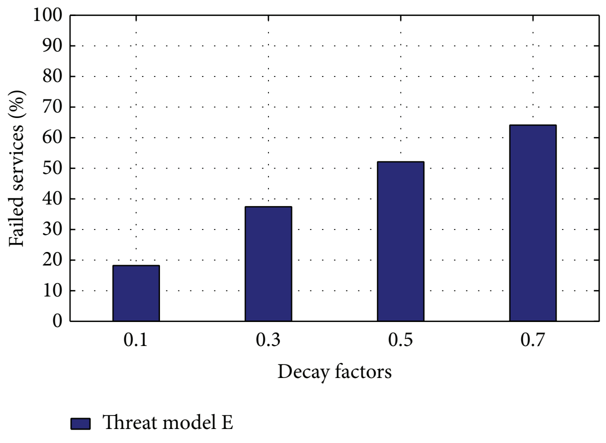


(b) t_LTCP = 0.005, t_STCP = 0.5

FIGURE 13: Performance comparison on all trust propagation models under threat model E, $d\% = 3.4\%$.

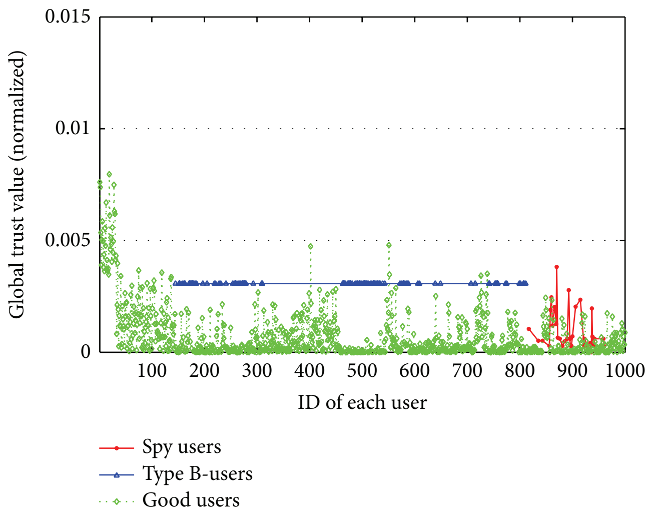


(a) STCP with different threshold values

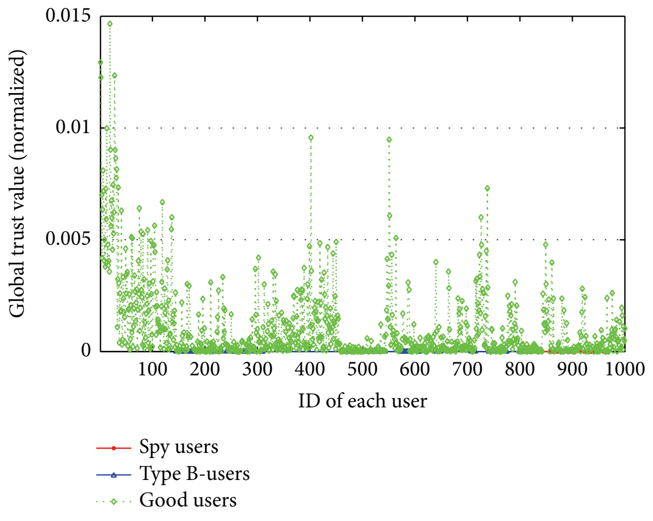


(b) DSLUTP with different decay factors

FIGURE 14: Impact of threshold values and decay factors under threat model E, $d\% = 3.4\%$, $h\% = 50\%$.



(a) STCP with threshold value = 0.2



(b) STCP with threshold value = 0.5

Figure 15: Global trust values of each user in STCP under threat model E, $d\% = 3.4\%$, $h\% = 50\%$.

or static threshold value, and should the threshold value be personalized or globalized.

Figure 14(b) shows the decay factors influence on the performance of DSLUTP. We observe that the lower the decay factor is, the better that the DSLUTP performs. Intuitively thinking, we should select the decay factor as low as we can. However, a too much low decay factor also weakens the trust propagation from good ones to good ones. The pretrust users will have too much higher global trust values than all of the rest, which is not good for a trust management system. We also observe that incorporating the decay factor only prevents or constrains the global trust values to be propagated to type-B malicious peers from pretrust peers and alleviates the detrimental effect of the malicious chain of type-B malicious collectives.

The second set of experiments shows the global trust value of each user with different threshold values (0.2 and 0.5) in STCP under threat model E with honest rating probability $h\% = 50\%$.

In Figure 15, we can see that, with a higher threshold value (0.5), the malicious users including spy users and malicious type-B users could hardly get any high global trust values to subvert the reputation management system. This also illustrates why our STCP with threshold value 0.5 performs very well under threat model E. But when the threshold value is 0.2, it cannot effectively cut the propagation way from good users to spy users. The malicious type-B users get much higher global trust values. This also illustrates why SLUTP and DSLUP and other uniform trust propagation models could not perform very well under threat model E.

6. Conclusions

We have discussed 10 kinds of trust propagation models from the simplest uniform trust propagation model (UTP1) to the most sophisticated conditional trust propagation model (DHCP) for reputation management systems. This paper compares these trust propagation models' attack resilience on 5 classic threat models and shows the different impact factors' influence on propagation models. First, we formulized all these trust propagation models mathematically and depict a small example trust network to intuitively show the performance of these propagation models with the global trust values of each user after the propagation process converge. Second, we found that by carefully introducing the similarity into the trust uniform and conditional propagation process, it could enhance the attack resilience of the reputation management system. We show that we could use similarity only as weight (SLUTP) or threshold value reference (STCP) or combine them together (HCP). The experiments on Epinion dataset show that if we should apply similarity both as weight and threshold value reference into the trust propagation process. Third, the experiments results also prove that decay factor also could alleviate the detrimental effect of malicious collude in both uniform trust propagation and conditional trust propagation. Experimental evaluation with five attack models shows that DHCP always performs the best on a real dataset.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is funded by Grants from National Nature Science Foundation of China (no. 61272173), Graduate Creative Talents Project of DUT for the remaining authors. The first author performed this work while visiting DiSL and School of CS in Georgia Institute of Technology, funded by China Scholarship Council. Any opinions, findings, and conclusions

or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] A. Y. Ng, A. X. Zheng, and M. I. Jordan, "Link analysis, eigenvectors and stability," in *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI '01)*, pp. 903–910, Seattle, Wash, USA, August 2001.

[2] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman, "Reputation systems," *Communications of the ACM*, vol. 43, no. 12, pp. 45–48, 2000.

[3] M. Srivatsa, L. Xiong, and L. Liu, "TrustGuard: countering vulnerabilities in reputation management for decentralized overlay networks," in *Proceedings of the 14th International Conference on the World Wide Web (WWW '05)*, pp. 422–431, ACM Press, New York, NY, USA, 2005.

[4] P. Resnick and R. Zeckhauser, "Trust among strangers in internet transactions: empirical analysis of eBay' s reputation system," *Advances in Applied Microeconomics*, vol. 11, pp. 127–157, 2002.

[5] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in P2P networks," in *Proceedings of the 12th International Conference on World Wide Web (WWW '03)*, pp. 640–651, Budapest, Hungary, May 2003.

[6] L. Xiong and L. Liu, "PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 843–857, 2004.

[7] N. Griffiths, "Task delegation using experience-based multi-dimensional trust," in *Proceedings of the 4th International Conference on Autonomous Agents and Multi agent Systems (AAMAS '05)*, pp. 489–496, Utrecht, The Netherlands, July 2005.

[8] Z. Y. Su, L. Liu, M. C. Li, and X. Fan, "ServiceTrust: trust management in service provision networks," in *Proceedings of the 10th International Conference on Services Computing (SCC '13)*, pp. 272–279, Santa Clara, Calif, USA, 2013.

[9] A. Abdul-Rahman and S. Hailes, "Supporting trust in virtual communities," in *Proceedings of the 33rd Annual Hawaii International Conference on System Siences (HICSS '00)*, Island of Maui, Hawaii, USA, January 2000.

[10] H. F. Liu, E. Lim, H. W. Lauw et al., "Predicting trusts among users of online communities: an epinions case study," in *Proceedings of the 9th ACM Conference on Electronic Commerce (EC '08)*, pp. 310–319, Chicago, Ill, USA, July 2008.

[11] J. Witkowski, "Trust mechanisms for online systems," in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011*, pp. 2866–2867, Barcelona, Spain, July 2011.

[12] G. Vogiatzis, I. MacGillivray, and M. Chli, "A probabilistic model fortrust and reputation," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pp. 225–232, Toronto, Canada, 2010.

[13] A. Josang and R. Ismail, "The beta reputation system," in *Proceedings of the 15th Bled Electronic Commerce Conference*, pp. 41–55, Bled, Slovenia, 2002.

[14] J. I. Andrew and J. Pitt, "On the classification of emotions and its relevance to the understanding of trust," in *Proceedings of the Workshop on Trust in Agent Societies, at the 10th International Conference on Autonomous Agents and Multi-Agent Systems*, pp. 69–82, New York, NY, USA, 2011.

[15] L. L. Jiang and M. Perc, "Spreading of cooperative behaviour across interdependent groups," *Scientific Reports*, vol. 3, article 2483, 2013.

[16] Q. Li, M. Y. Chen, M. Perc, A. Iqbal, and D. Abbott, "Effects of adaptive degrees of trust on coevolution of quantum strategies on scale-free network," *Scientific Reports*, vol. 3, article 2949, 2013.

[17] Y. Z. Ren, M. C. Li, Y. Xiang, Y. R. Cui, and K. Sakurai, "Evolution of cooperation in reputation system by group-based scheme," *Journal of Supercomputing*, vol. 63, no. 1, pp. 171–190, 2013.

[18] M. Perc and P. Grigolini, "Collective behavior and evolutionary games—an introduction," *Chaos, Solitons & Fractals*, vol. 56, pp. 1–5, 2013.

[19] P. Lawrence, B. Sergey, M. Rajeev, and W. Terry, "The PageRank citation ranking: bringing order to the web," Tech. Rep., Stanford Info Lab, 1999.

[20] U. Kuter and J. Golbeck, "Using probabilistic confidence models for trust inference in web-based social networks," *ACM Transactions on Internet Technology*, vol. 10, no. 2, article 8, 2010.

[21] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of trust and distrust," in *Proceedings of the 13th International World Wide Web Conference (WWW '04)*, pp. 403–412, New York, NY, USA, May 2004.

[22] F. J. Ortega, J. A. Troyano, F. L. Cruz, C. G. Vallejo, and F. Enríquez, "Propagation of trust and distrust for the detection of trolls in a social network," *Computer Networks*, vol. 56, no. 12, pp. 2884–2895, 2012.

[23] Y. Wang and J. Vassileva, "Trust and reputation model in peer-to-peer networks," in *Proceedings of the 3rd International Conference on Peer-to-Peer Computing (P2P '03)*, pp. 150–157, Linköping, Sweden, 2003.

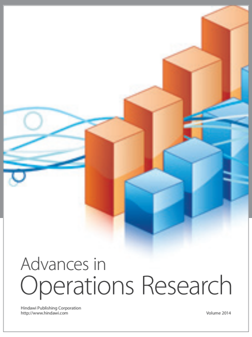
[24] S. Buchegger and J. L. Boudec, "A robust reputation system for P2P and mobile ad-hoc networks," in *Proceedings of the 3rd Workshop on the Economics of Peer-to-Peer Systems*, Cambridge, Mass, USA, 2004.

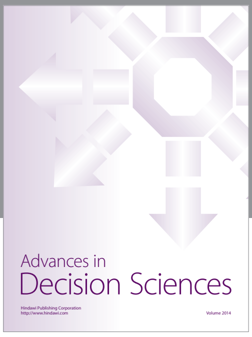
[25] S. Song, K. Hwang, R. F. Zhou, and Y. Kwok, "Trusted P2P transactions with fuzzy reputation aggregation," *IEEE Internet Computing*, vol. 9, no. 6, pp. 24–34, 2005.

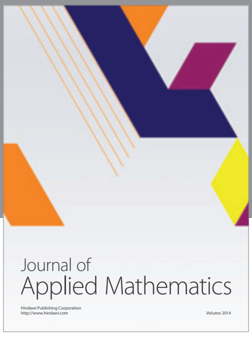
[26] J. Weng, C. Miao, and A. Goh, "Improving, collaborative filtering with trust-based metrics," in *Proceedings of the ACM Symposium on Applied Computing (SAC '06)*, pp. 1860–1864, New York, NY, USA, April 2006.

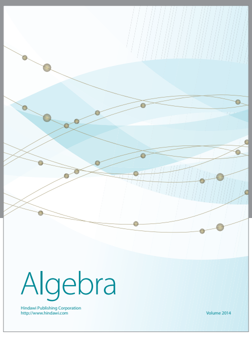
[27] P. Massa and P. Avesani, "Trust-aware bootstrapping of recommender systems," in *Proceedings of the 17th European Conference on Artificial Intelligence*, pp. 29–33, Riva del Garda, Italy, 2006.

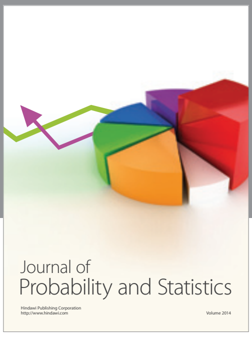
[28] T. C. Peng and S. T. Chou, "iTrustU: a blog recommender system based on multi-faceted trust and collaborative filtering," in *Proceedings of the ACM Symposium on Applied Computing (SAC '09)*, pp. 1278–1285, Honolulu, Hawaii, USA, March 2009.











Hindawi

Submit your manuscripts at
http://www.hindawi.com