# Trust Evaluation in Online Social Networks Using Generalized Network Flow

Wenjun Jiang, Jie Wu, *Fellow, IEEE*, Feng Li, *Member, IEEE*, Guojun Wang, *Member, IEEE*, and Huanyang Zheng

**Abstract**—In online social networks (OSNs), to evaluate trust from one user to another indirectly connected user, the trust evidence in the trusted paths (i.e., paths built through intermediate trustful users) should be carefully treated. Some paths may overlap with each other, leading to a unique challenge of *path dependence*, i.e., how to aggregate the trust values of multiple dependent trusted paths. OSNs bear the characteristic of high clustering, which makes the path dependence phenomenon common. Another challenge is *trust decay* through propagation, i.e., how to propagate trust along a trusted path, considering the possible decay in each node. We analyze the similarity between trust propagation and network flow, and convert a trust evaluation task with path dependence and trust decay into a generalized network flow problem. We propose a modified flow-based trust evaluation scheme *GFTrust*, in which we address path dependence using network flow, and model trust decay with the leakage associated with each node. Experimental results, with the real social network data sets of Epinions and Advogato, demonstrate that GFTrust can predict trust in OSNs with a high accuracy, and verify its preferable properties.

**Index Terms**—Generalized network flow, online social networks (OSNs), path dependence, trust decay, trust evaluation

## 1 Introduction

"To be trusting is to be fooled from time to time; to be suspicious is to live in constant torment (Wu [1])." People face trust issues every day in real life. The trust mechanism is a tool used to facilitate decision making in diverse applications. This paper copes with the setting in which a source $s$ is interested in a single target $d$ (it can be a person, or a product/service he provides) in online social networks (OSNs). Some users have preconceived opinions about $d$. $s$ might desire to estimate whether or not she would like $d$, based on the aggregated opinions of others. In real life, $s$ might first consult her friends for their recommendations. In turn, the friends, if they do not have opinions of their own, may consult their friends, and so on. Based on the cumulative feedback $s$ receives, she can form her own subjective opinion. A trust evaluation system aims to provide a similar process to produce high-quality trust prediction for users.

"Trust in a person is a commitment to an action, based on a belief that the future actions of that person will lead to a good outcome (Golbeck [2])." Trust can be built through direct contact (first-hand), such as a directed link from $s$ to $u$, or through a recommendation (second-hand), such as a *trusted path* $(s, u, d)$ representing $s$'s trust of $d$ via $u$'s recommendation (Fig. 1a). A path is constructed through iterative recommendations. Multiple sequential and parallel paths are overlapped to form a directed *trusted graph* from $s$ to $d$. In Fig. 1a, $e(s, u)$ and $e(u, d)$ are two edges of a sequential path $(s, u, d)$; $(s, u, d)$ and $(s, v, d)$ are two parallel paths; $(s, v, u, d)$ is an overlapped path with paths $(s, u, d)$ and $(s, v, d)$. Usually, each edge has a weight value between 0 (no trust) and 1 (full trust) to quantify each direct trust.

- W. Jiang is with the School of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan Province 410082, P. R. China. E-mail: jiangwenjun@hnu.edu.cn.
- J. Wu and H. Zheng are with the Department of Computer and Information Sciences, Temple University, 1925 North 12th Street, Philadelphia, PA 19122. E-mail: jiewu@temple.edu, huanyang.zheng@gmail.com.
- F. Li is with the Department of Computer and Information Technology, Indiana University-Purdue University, Indianapolis, IN 46202. E-mail: fengli@iupui.edu.
- G. Wang is with the School of Information Science and Engineering, Central South University, Changsha, Hunan Province 410083, P. R. China. E-mail: csgjwang@csu.edu.cn.

*Manuscript received 8 June 2013; revised 20 Apr. 2015; accepted 2 May 2015. Date of publication 19 May 2015; date of current version 10 Feb. 2016.*
*Recommended for acceptance by F. Rodríguez-Henríquez.*
*For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.*
*Digital Object Identifier no. 10.1109/TC.2015.2435785*

### 1.1 The Motivation

*Trust aggregation* is still an open problem, although several attempts have been made (Golbeck [2], Sun et al. [3], Wang and Wu [4], Jiang et al. [5], Mahoney et al. [6], Taherian et al. [7]). Two open challenges are "how to aggregate the trust of overlapped paths" and "how to calibrate trust decay over iterative recommendations." Some used high-level aggregation rules (Sun et al. [3]), including the sequential rule, where concatenation propagation of trust does not increase trust (i.e., the trust of $(s, u, d)$ is no more than that of $e(s, u)$ or $e(u, d)$), and the parallel rule, where multi-path propagation of trust does not reduce trust. However, these rules are too general to provide specific calculation guidance for trust aggregation. Moreover, most of the existing work cannot solve the two challenges simultaneously. OSNs bear the small-world characteristic of high clustering (Watts [8]). Because of this, the path dependence phenomenon is much more common in OSNs. Our motivation is to develop an efficient scheme that can address the above two challenges simultaneously, and provide guidelines for automatic trust prediction in OSNs.

Trust propagation is similar to a flow passing process. The amount of flow corresponds to the amount of trust.

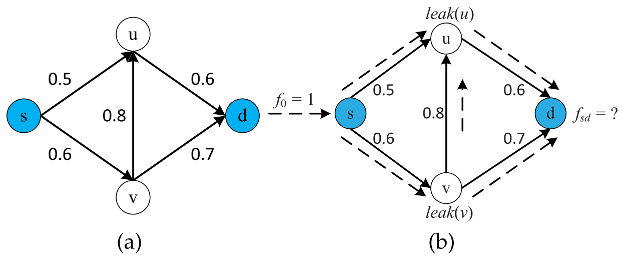


Fig. 1. (a) An example of a trusted graph; (b) Evaluating trust using GFTrust, in which the dashed arrow line represents trust flow.

Initially, the source is given a certain amount of trust to be allocated. Then a flow passes from the source to the sink corresponds to the trust that is being propagated in a trusted path. Total trust is an aggregation through different paths (flows) from the source to sink. As network flows can be split and merged, we address the path dependency challenge. In addition, a flow may leak during its passing, which can be used to address the trust decay challenge.

Besides solving the two challenges, we also strive to design a unique trust evaluation system that is incentive compatible, friendly to newcomers, and tolerant to the malicious behavior of making Sybils.

### 1.2 Main Ideas

We propose a novel computational approach for calculating trust, based on a modified network flow model with leakage. As shown in Fig. 1b, given an initial flow of 1 (i.e., $f_0 = 1$ representing full trust) at $s$, what is the final flow (i.e., $f_{sd}$ representing total aggregated trust) that $d$ can get? This flow model addresses path dependence by allowing flow split and merge at each node. Moreover, it is commonly agreed upon that people place more weight on direct contacts than on indirect contacts. Therefore, any viable trust model must address trust decay over iterative recommendations. We introduce a notion of leakage associated with each recommendation node, which is analogous to a leakage in a water pipe. At each node other than $s$ and $d$, a certain percentage of incoming flow will be leaked before redirecting to outgoing links (e.g., $leak(u)$ and $leak(v)$ in Fig. 1b). We will describe the calculation details later.

### 1.3 Our Contributions

We propose the *GFTrust* scheme, where we introduce a modified generalized network flow (simply generalized flow) model (Wayne [9]) to cope with the trust evaluation task with path dependence and trust decay (Fig. 1b). Once the trust from $u$ to $v$ passes a given threshold, $v$ is taken as trusted by $u$, but with a given limit on capacity. The full capacity is 1, corresponding to a full trust. Our model is analogous to a credit card system, where $s$ takes the role of a bank; each neighbor is a credit card owner who is allowed to use a card, but with a given credit limit. Other nodes can also be taken as a bank or credit card owner in the same way. Each indirect reference corresponds to a percentage of credit loss. The goal is to decide whether or not $s$ can accept the application of $d$, by calculating the proper credit that $d$ receives. Our contributions are threefold:

1) Our work is the first to address the two challenges of path dependence and trust decay simultaneously, in the domain of trust evaluation in OSNs. Also, we use a modified generalized flow model with leakage, which is a novel approach in trust evaluation.
2) As a flow-based model, GFTrust has the advantage of generality, while saving the normalization process (since the resulting trust will never be larger than 1). Moreover, it bears the properties of incentive compatibility and Sybil tolerance, and it coincides with the basic axioms that a trust model should meet (as shown in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TC.2015.2435785).
3) We conduct extensive experiments on a real social network data set of Epinions (Jiang et al. [5]). Some more experiments on the data set of another social network Advogato (Levien [10]) is shown in Appendix B, available in the online supplemental material. The experimental results validate the effectiveness of GFTrust, and also verify its preferable properties. Moreover, the use of flow increases the accuracy of trust prediction, while the leakage decreases the deviation between the values of calculated trust and the direct trust.

The remainder of this paper is organized as follows: Section 2 analyzes the background of the two challenges, and surveys related work in the literature. Section 3 states the problem we address, and provides the preliminary concepts we will use in network flow theory. Sections 4 and 5 present the solution overview and the algorithm details. Section 6 analyzes the features and properties. Section 7 describes the experimental evaluation. Finally, Section 8 concludes this paper and suggests future work.

## 2 Background and Related Work

In this section, we first analyze the necessity of addressing the two challenges of path dependence and trust decay. Then, we briefly review the literature.

### 2.1 Path Dependence

Some models can deal with path dependence (e.g., Golbeck [2], Jøsang et al. [11]), but they may cause information loss or reuse. Taking the trusted graph in Fig. 1a for example, previous models go to two extremes: (1) some ignore the overlapped edge $e(v, u)$, by using only the shortest paths $(s, u, d)$ and $(s, v, d)$ (Lin et al. [12]) or ignore more information by only considering the shortest, strongest paths (Golbeck [2]), which will lead to the loss of information; (2) others take $(s, v, u, d)$ as an independent path (Jøsang et al. [11]) by using all three paths, which will reuse the information of $e(s, v)$ and $e(u, d)$.

### 2.2 Trust Decay

Let us consider a scenario where $s$ fully trusts $v_1$, and $v_i$ fully trusts $v_{i+1}$, $i \in [2, n-1]$, and finally $v_n$ fully trusts $d$. Then, how about the level in which $s$ trusts $d$? Two commonly used methods will calculate trust as follows: (1) Multiplication. $t(s,d) = t(s,v_1) \cdot \prod t(v_i, v_{i+1}) \cdot t(v_n, d) = 1$. (2) Taking the minimum. $t(s,d) = \min\{t(s,v_1), t(v_i, v_{i+1}), \ldots,$

$t(v_n, d)\} = 1$. The result will be that $s$ will fully trust many indirectly connected users who are far away from him/her, which is inconsistent with real life. This indicates the strong necessity of considering trust decay through propagation. Some models have mentioned trust decay (e.g., Golbeck [2], Sun et al. [3], Jøsang et al. [11]). However, they are on a coarse-grained level, i.e., they can only guarantee that trust does not increase during propagation.

### 2.3 Related Work

The problem of path dependence arises when combining information from multiple sources that include unknown amounts of correlation, which have been studied in the trust domain (Jøsang et al. [11]), and other domains including information fusion, decision fusion, and control theory. These areas use mutual information (Paninski [13]), entropy calculation or Bayesian analysis (Bailey et al. [14], Chen and Varshney [15]), and state space analysis (Friedland [16]) as tools. The phenomenon of decay usually emerges with distance, which also has been mentioned (but not well addressed) (Golbeck [2], Jøsang et al. [11], Ziegler and Lausen [17]). For instance, trust decay is inherent to Appleseed (Ziegler and Lausen [17]), by setting a spreading factor. Recognizing the similarity between trust propagation and flows, we use a natural way to model and tackle the two challenges using network flows in the domain of OSNs.

Network flow theory has been used in many fields (Ahuja et al. [18]), and has recently been introduced into a trust evaluation system. Given capacity limits on the edges, the goal of the maximum flow problem is to send as much flow as possible from the source to the sink (see Ahuja et al. [18] for details). Applying this idea, Levien [10] proposes the Advogato maximum flow trust metric to distinguish vicious nodes from good nodes. Given the network structure, Advogato takes some predefined trusted nodes as seeds, assigns them capacities, and outputs a set of trusted nodes that can gain flow from seeds; other nodes are taken as vicious, and will be excluded. Wang and Wu [4] measure trust using the network flow theory. Three FlowTrust algorithms are proposed to normalize trust metrics. However, none of existing works have explicitly addressed the two challenges of path dependence and trust decay simultaneously. Moreover, existing flow-based trust models have to normalize the maximum flow into a trust value in order to make it in a predefined range.

The generalized flow problem is a natural generalization of the traditional network flow. GFTrust uses a carefully designed generalized flow, which makes it able to solve the two challenges, save the normalization process, and bear some additional good properties. We will briefly describe the idea of generalized flow in the next section.

## 3 Problem Definition and Preliminary Concepts

In this section, we formulate the problem we address, and provide some preliminary concepts. The notations used in this paper are described in Table 1. Note that, $e(u,v)$, $g(u,v)$ (and other similar representations), are denoted as $e, g(e)$, when it is unnecessary to distinguish an edge from others.

TABLE 1
Notations

| SYMBOL | DESCRIPTION |
|---|---|
| $G = (V, E)$ | a trusted graph |
| $s/d$ | the source/destination |
| $u/v/u'/v'$ | a node in the trusted graph |
| $e(u,v)$ | an edge from node $u$ to node $v$ |
| $t(u,v)$ | trust value from node $u$ to node $v$ |
| $g(u,v)$ | the gain factor of edge $e(u,v)$ |
| $c(u,v)$ | the capacity of edge $e(u,v)$ |
| $f(u,v)$ | the flow of edge $e(u,v)$ |
| $f_r(v,u)$ | the flow of $e(v,u)$ in the residual network |
| $c_f(u,v)$ | the residual capacity of edge $e(u,v)$ |
| $g_f(v,u)$ | the gain factor of $e(v,u)$ in the residual network |

### 3.1 Problem Definition

Given a trusted graph $G = (V, E)$, $V$ is the set of nodes and $E$ is the set of edges. For two indirectly connected nodes, $s$ and $d$ in $V$, $s$ is the source and $d$ is the destination. For the confidence of user interactions in online social networks, we seek to determine how to design an efficient scheme to evaluate the trust level of $d$ for $s$; specifically, how are we to address the two challenges of path dependence and trust decay simultaneously?

In OSNs, trust evidence can be collected from three main sources (Sherchan et al. [19]): attitudes, behaviors, and experience. Trust has been represented in very different ways, such as continuous or discrete numerical values (e.g., Taherian et al. [7], Richardson et al. [20], Jøsang et al. [11], Abdul-Rahman and Hailes [21]), or probability/entropy (e.g., Sun et al. [3]). In this work, we assume that the direct trust values between any two connected users are already known, which are represented by continuous numerical values in [0,1], with 1 representing full trust (upper bound) and 0 representing no trust (lower bound). Our goal is to infer indirect trust values for any two unconnected users, based on the known ones.

### 3.2 Preliminary Concepts

The generalized network flow problem (Wayne [9]) is an extension of standard network flow, in which flow leaks as it is sent through the network. It is represented by a gain function in each edge, $g : E \rightarrow R$, where $R$ is the set of real numbers. For each unit of flow that enters an edge $e(u,v)$ at node $u$, $g(u,v)$ units will arrive at node $v$. A generalized flow should satisfy the capacity constraints: $0 \leq f(u,v) \leq c(u,v)$, and the generalized antisymmetry constraints: $\forall e(u,v) \in E : f(u,v) = -g(v,u) \cdot f(v,u)$, where $g(v,u) = -1/g(u,v)$, and the minus sign means that the flow is going in the opposite direction. Most of the existing network flow algorithms (Wayne [9]) are based on the Ford-Fulkerson method (Ford and Fulkerson [22]), the two key concepts of which are *residual network* and *augmenting path*.

**Definition 1 (Residual network).** *Let $f$ be a flow in a network $N = (V, E)$, where $c(u,v)$ and $g(u,v)$ are the capacity and gain factor of edge $e(u,v) \in E$, respectively. With respect to the flow $f$, the residual network is $N_f = (V, E_f)$, in which the residual capacity is defined by $c_f(u,v) = c(u,v) - f(u,v)$.*

**Definition 2 (Augmenting path).** *An augmenting path is a path in the residual network, where the capacity on each edge is larger than 0. A new flow can pass through an augmenting path.*

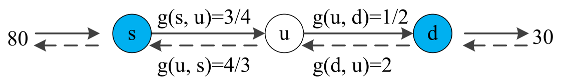


Fig. 2. An example of the generalized flow network.

Fig. 2 shows an example of a generalized flow network. When a flow of $f = 80$ enters into $s$, only $80 \cdot g(s, u) = 80 \cdot 3/4 = 60$ can go out of $u$, and $60 \cdot g(u, d) = 60 \cdot 1/2 = 30$ can go out of $d$. Suppose that $c(s, u) = c(u, d) = 100$. After sending the above flow $f$, the residual capacity will be $c_f(s, u) = 100 - 80 = 20$, $c_f(u, d) = 100 - 60 = 40$, both of which are larger than 0, then $(s, u, d)$ is an augmenting path.

## 4 Solution Overview

We give the overview of GFTrust. First of all, since people join in OSNs because they hope to collaborate and interact with others, we assume that everyone starts cooperatively and trusting in others in the absence of feedback (we call it the "initial trust assumption"). And as time passes by, that trust will be shaped according to the real experience. Biologists verified that cooperation with trust is our instinct as human beings through natural selection (Nowak [23], Manapat et al. [24], Rand et al. [25]); this can serve as convincing evidence of our assumption. Based on this, we deem the trust evaluation process for two indirectly connected users, $s$ and $d$, as follows: at first, $s$ fully trusts $d$. Then, according to the friends' suggestions or comments, the initial trust shrinks little by little. We then model the process using network flow, with three tasks as follows:

*Task 1: Determine the initial flow*. According to the initial trust assumption, we let the initial flow from $s$ be $f_0 = 1$. We find that 1 is the most proper and natural value of initial flow. Let us try other settings. If $f_0 = 0$, then there is no flow to send; in this case, no matter how good of a recommendation the intermediate nodes make, $s$ will not trust $d$ at all. In another case of $f_0 = \infty$, there is infinite flow to send; as long as there are enough paths from $s$ to $d$, $s$ will highly trust $d$, since a large flow (even $> 1$) will reach $d$. Therefore, both settings are not suitable for our scenario. A more flexible alternative is to let $0 < f_0 < 1$. However, there is no need to adjust it, since the node leakage will take the same role. That is, we can set a larger node leakage if $f_0$ is larger, and vice versa. So, we fix $f_0 = 1$, and adjust the node leakage.

*Task 2: Explore the node leakage*. We set the leakage to be associated with nodes, since trust decay is caused by nodes instead of edges. Just mentioning one example, the direct trust from $s$ to $u$, $t(s, u)$, will not decay through trust propagation from $s$ to $d$ via $u$. For modeling node leakage, the most challenging issue is "How much flow should leak in each node?" In real life, it may be very complex since many factors may impact the answer, such as the distance from source, the tie strength between users, and the personality of users (some people may opt to trust others, while some others may opt to distrust). Currently, we mainly consider the factor of distance (from the source). The GFTrust scheme offers a framework for considering and integrating other factors in a reasonable way for the future.

Moreover, we take the approach of the proportional leakage, in which trust (flow) will shrink a certain proportion during its propagation in each intermediate node. Another possible method is the fixed value leakage, where trust (flow) will lose a certain fixed amount. Intuitively, both of the two approaches make sense. Here, we would like to put the latter into the future work, and only consider the former.

*Task 3: Assign the edge capacity*. According to our application scenario, we use the trust value $t(e)$ on edge $e$ to represent its capacity, which limits the maximum flow (trust) that can pass through the edge. In this way, the trust value on each edge cannot be overused. Thus it can avoid reusing some information, especially when there exist dependent paths.

Based on the above analysis, we design three key steps for GFTrust: (1) Modeling trust decay with node leakage; (2) Constructing generalized flow network; and (3) Calculating a near-optimal generalized flow.

There are two reasons for setting the goal of deriving a near-optimal generalized flow: (1) It is still an open problem of calculating the generalized maximum flow in polynomial time complexity. According to (Wayne [9]), the time complexity in the worst-case is $O(nm(m + n\ lg\ n)lg\ B)$, where $n$ is the number of nodes, $m$ is the number of edges, and $B$ is the biggest integer for scaling decimal numbers to integers. Kevin (Wayne [9]) presented a family of $\xi$-approximation algorithms for every $\xi > 0$, which can improve the above complexity by a factor of $m$. Perillo and Heinzelman [26] uses generalized maximum flow to maximize the lifetime of energy-constrained wireless sensors, in which the optimal flow is calculated with linear programming. Its complexity is $O(p^3 L)$, where $p$ is the number of variables, and $L$ is the variable resolution. (2) A near-optimal maximum flow is acceptable in our scenario, because the task of trust prediction is to estimate a trust level which is close to the direct trust (i.e., the ground truth or the expressed trust), instead of maximizing a trust value.

## 5 GFTrust: the Algorithm Details

In this section, we introduce the details of GFTrust. Since many works of trust evidence collection have been done (e.g., Golbeck [2], Massa and Avesani [27]), we do not focus on how to collect the information. Without loss of generality, we assume that the trusted graph is already known, i.e., the trust relationships and values of directly-connected neighbors are already available.

### 5.1 Modeling Trust Decay with Node Leakage

As mentioned before, trust may decay during its propagation in a trusted path. We design a series of leakage functions to simulate their patterns (Fig. 3).

A simple scheme is the uniform leakage (Fig. 3a), where trust decays with the same percentage in each intermediate node $v \in V \setminus \{s, d\}$. Intuitively, the leakage cannot be too large, and we try some values in the experiments, as shown in Eq. (1).

However, trust may decay differently along the propagation; it may first decay quickly (a larger decay), then slowly (a smaller decay), or vice versa. Therefore, we also consider the non-uniform leakage, i.e., the percentage of leakage varies among different intermediate nodes, according to their distance from the source. Three types

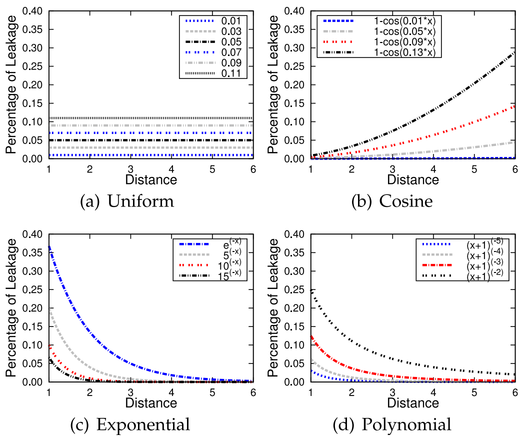


Fig. 3. Four types of leakage functions.

of mathematical functions have been examined, as shown in Eqs. (2), (3), and (4):

$$\lambda_1(x) = leak, leak \in [0, 0.5), \tag{1}$$

$$\lambda_2(x) = 1 - \cos(kx), k \in \{0.01, 0.05, 0.09, 0.13\}, \tag{2}$$

$$\lambda_3(x) = l^{(-x)}, l \in \{e, 5, 10, 15\}, \tag{3}$$

$$\lambda_4(x) = (x+1)^m, m \in \{-2, -3, -4, -5\}, \tag{4}$$

where $x$ is the distance from the current intermediate node to the source. Other parameters $k, l, m$ are set tentatively. The listed values can lead to a reasonable leakage range of $[0, 0.4)$ (Fig. 3).

Since it is not clear how trust will decay during its propagation, we design the above four leakage functions to imitate four possible decay patterns. Some of the functions leak more at the closer nodes, while others leak more at distant nodes. Actually, these functions all favor the shorter paths over the longer ones. We will test their effects in the experiments.

## 5.2 Constructing Generalized Flow Network

To conduct trust evaluation using a generalized flow algorithm, we should first construct the generalized flow network, for which the capacity and gain factor of each edge should be set. We design Algorithm 1 for the process. As mentioned before, the capacity is set using the trust value. Then, the main task is to cope with the gain factor, for which we have two principles:

1) For the outgoing edges from $s$ and the incoming edges to $d$, since all the flows going on these edges will be fully passed on, we let their gain factors be 1.
2) For the intermediate nodes where flows will leak, the gain factors should be less than 1. Moreover, in the generalized flow algorithms, the gain factor is associated with edge. Thus, we take two steps to transform the leakage of a node into the gain factor of an edge (an example is shown in Fig. 4):

*Step 1*: splitting each intermediate node (lines 1-2). To be specific, we split a node $v$ into two nodes of $v^+$ (+ indicates flow coming into it) and $v^-$ ($-$ indicates flow going out from it). Then, we add a new edge $e(v^+, v^-)$ into the graph, which is called intermediate edge, corresponding to the

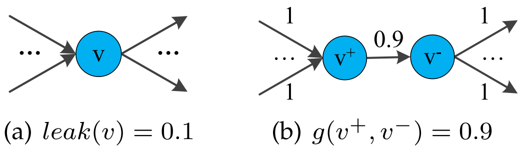


Fig. 4. Transforming node leakage to edge gain factor.

intermediate node. The incoming neighbors of $v$ are linked to $v^+$, and outgoing neighbors to $v^-$.

**Algorithm 1.** Transform($G, s, d$)

```
Input: $G$, a trusted graph; $s$, source; $d$, destination.
Output: $G'$, a generalized flow network.
1: for each intermediate node $v$ in $G$ do
2:     Split $v$ into $v^+$ and $v^-$; Add an edge $e(v^+, v^-)$;
       $N_{out}(v^-) \leftarrow N_{out}(v)$; $N_{in}(v^-) \leftarrow N_{in}(v)$.
3: for each edge $e$ in $G$ do
4:     if $e$ is an intermediate edge then
5:         $c(e) \leftarrow 1$; $g(e) \leftarrow 1 - leak(v)$.
6:     else
7:         $c(e) \leftarrow t(e)$; $g(e) \leftarrow 1$.
```

*Step 2*: assigning the gain factor for the edge according to the node leakage. First, we model the trust value of an edge as its capacity (lines 3-7). Note that we let $t(v^+, v^-) = 1$. The intuition is that a user will always trust his own opinion. Accordingly, $c(v^+, v^-) = 1$. The gain factor of an intermediate edge $e(v^+, v^-)$ is $g(e) = 1 - leak(v)$ (lines 4-5), while that of other edges is 1 (lines 6-7). The percentage of $leak(v)$ can be set according to Eqs. (1), (2), (3), (4).

## 5.3 Calculating Near-Optimal Generalized Flow

The goal of GFTrust is to solve the two challenges of path dependence and trust decay, and predict a trust level that is close to the truth. Therefore, we design a near-optimal maximum flow algorithm. Before introducing the whole process, we describe the following two observations:

Observation 1. In the original trusted graph, a shorter trusted path makes a higher gain, i.e., a flow passing on a shorter path remains a larger amount than that passing on a longer path.

Consider two paths: $q_1 = \{s, v_1, \ldots, v_m, d\}$ and $q_2 = \{s, u_1, \ldots, u_m, \ldots, u_n, d\}$, $n > m$. Suppose a flow $f$ meets the capacity constraints. Then, when $f$ passes on $q_1$, the resulting flow will be $f_1 = f \cdot \prod_{i \in [1,m]} (1 - leak(v_i))$. Similarly, the same $f$ passing on $q_2$ will result in $f_2 = f \cdot \prod_{i \in [1,n]} (1 - leak(v_i))$. We can compare $f_1$ and $f_2$ through the result of $\xi = \frac{f_2}{f_1}$. Since $leak(v_i) \in [0, 1)$, then $\xi = \prod_{i \in [m+1,n]} (1 - leak(v_i)) \leq 1$. Therefore, we have $f_2 \leq f_1$. So, we complete the proof.

Taking Fig. 5 for instance, suppose the leakage is 0.1 in each intermediate node; then a flow $f = 0.6$ will become $0.6 \cdot 0.9 = 0.54$ through the upper path $q_1 = (s, v_1^+, v_1^-, d)$, and $0.6 \cdot 0.9 \cdot 0.9 = 0.486$ through the lower path $q_2 = (s, v_2^+, v_2^-, v_3^+, v_3^-, d)$. As an extension, if $f > 0.6$, then sending 0.6 through $q_1$ and the remainder through $q_2$ will get a larger flow than that in the opposite order.

Observation 2. In the original trusted graph, the trusted paths with the same length have the same efficiency of sending flow.

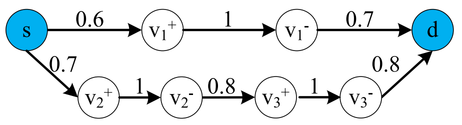


Fig. 5. An example for Observation 1.

Consider a special case of Observation 1: $m = n$. Then we can get $f_1 = f_2$. That is, as long as a flow meets the capacity constraints, sending it through any path with the same length will lead to the same result.

Based on the above two observations, we propose a greedy algorithm to select the shortest trusted path and augmenting flow via the path (Algorithm 2). Fig. 6 shows an example of the calculation process.

**Algorithm 2.** GFNearOptimal($G'$, $s$, $d$)

```
Input: $G'$, a generalized flow network.
Output: $f^*$, a near-optimal flow.
 1: Initialize $f(s) \leftarrow 1$; $f^\Delta \leftarrow 0$.
 2: while $f(s) > 0$ do
 3:     Search the shortest augmenting path $p$.
 4:     return $f^*$ if $p = \varnothing$.
 5:     for each edge $e$ in $p$ do
 6:         $f \leftarrow min\{f(s), f \cdot g(e), c(e)\}$.
 7:     $f^* \leftarrow f^* + f$.
 8:     for each edge $e$ from $d$ to $s$ do
 9:         $f_r \leftarrow f \cdot 1/g(e)$; $c(e) \leftarrow c(e) - f_r$.
10:     $f(s) \leftarrow f(s) - f_r$.
```

### 5.3.1 Searching the Shortest Path in a Trusted Graph

Based on the above analysis, we adapt the idea of the Edmonds-Karp algorithm (Edmonds and Karp [28]), using the breadth-first search to find the shortest trusted path, which will be used as an augmenting path. Taking Fig. 6 for instance, let $p_1 = (s, u^+, u^-, d)$, $p_2 = (s, v^+, v^-, d)$, $p_3 = (s, v^+, v^-, u^+, u^-, d)$. The process of selecting a path is: (1) do the breadth-first search and find the first unused shortest trusted path, suppose it is $p_2$; (2) send flow from $s$ to $d$ through $p_2$. After that, record $p_2$ in a used path list. Note that the same path will not be used again. But, some edges of it may be used more than once when they are included in some other paths. However, as we have mentioned before, the capacity on each edge is exactly the trust value, and at most all the capacity is used up. Therefore, it overcomes the drawback of information reuse in some existing models, which may lead to inaccurate trust evaluation results.

### 5.3.2 Augmenting Flow

We iteratively take the shortest trusted path, and execute the following two operations (Algorithm 2), until one of the two end conditions is met: (1) there is no flow remaining to be sent (i.e., $f(s) = 0$, line 2), or (2) there are no augmenting paths (i.e., $p = \phi$, line 4) for any flow to pass.

*Operation 1:* Augmenting a flow $f$ through the selected path. Let $f(s)$ be the amount of flow $s$ can send out, and $e \in E$ be an edge in the path; $f$ should satisfy: $f \leq c(e)$, and $f \leq f(s)$. More importantly, $f$ leaks in each intermediate edge, denoted as $f = f \cdot g(e)$. Iteratively do this until $f$ reaches $d$.

*Operation 2:* Calculating the residual capacity of each edge from $d$ to $s$, as well as the residual flow that $s$ can send out. We do this by iteratively subtracting flow $f_r$ (corresponding flow of $f$ in the residual network) from capacity, from $d$ to $s$. The flow in the residual network is $f_r = -f/g(e)$. The residual capacity is $c_f(e) = c(e) - f_r$, and the residual flow of $s$ is $f(s) = f(s) - f_r$.

Since $f_0 = 1$ and flow leaks when it passes through trusted paths, the result of Algorithm 2 is in the range of $[0, 1]$, which is the same with the trust value. Therefore, the resulting flow can be taken as a trust value directly, without normalization.

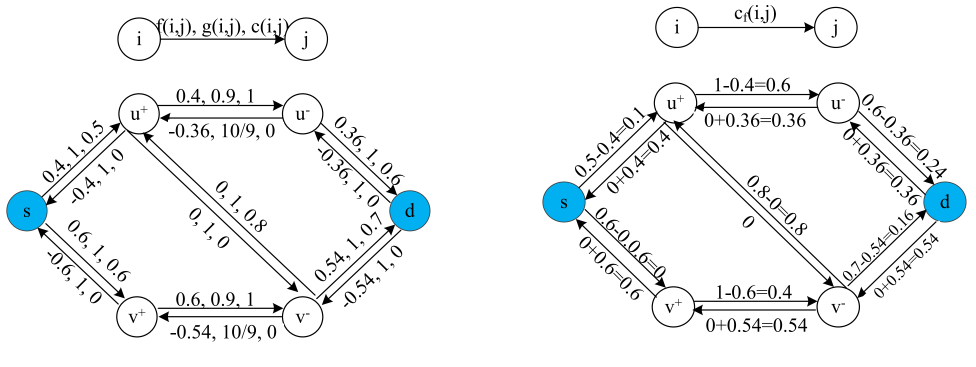
## 6 Analysis of GFTrust

We give an extensive analysis of GFTrust: its efficiency and near-optimal effect, its unique advantages, its incentive compatibility, and its malicious behavior resistance properties. We also analyze its conformity with basic axioms in Appendix A, available in the online supplemental material.

## 6.1 Efficiency and Near-Optimal Effect

### 6.1.1 Efficiency

From the example in Fig. 6, we can see that only two paths are used to send out flows, i.e., the iterative number of augmenting flows is $k = 2$. We observe that in most cases, the



(a) Generalized network (b) Residual network

Fig. 6. The process of calculating a feasible flow for the example trusted graph in Fig. 1a. Steps: (1) Find the shortest trusted path $(s, v^+, v^-, d)$ to send flow. It results in $f_1 = 0.54$, and the residual flow of $s$ is $1 - 0.6 = 0.4$. (2) Send the residual flow along the second shortest trusted path $(s, u^+, u^-, d)$, which results in $f_2 = 0.36$. With $f(s) = 0$, there is no flow left to be sent. (3) The near-optimal generalized flow is $f^* = f_1 + f_2 = 0.9$.

iterative number in GFTrust is quite small and can be taken as a constant.

*Observation 3*. In most cases, the iterative number of augmenting flows in GFTrust is a small constant.

The trick lies in the limited initial flow and the selection of augmenting path. The total flow to be sent is small, i.e., $f_0 = 1$. Each time we augment flow, we select a trusted path, whose minimum capacity should be larger than a predefined threshold. Given the range of trust value $[0, 1]$, the threshold is usually at least as large as half trust, i.e., 0.5. Even when we set the threshold to be 0.1, then the capacities of edges in trusted paths are not smaller than 0.1, which indicates that at least a flow of $f = 0.1$ can be sent out through a path. Therefore, all the initial flow can be sent out through, at most, 1/0.1 = 10 paths. In summary, the iterative number of augmenting flow is small, because the flow supply is small and the "pipe" is wide.

However, in some special cases, when there are not enough wide "pipes," the problem is as complex as the standard maximum flow problem, for which, the iterative number of augmenting flows is $O(|V||E|)$ (Cormen et al. [29]). Note that, although a small number of trusted paths may be used, all the trusted paths have been considered in GFTrust, i.e., all paths have the chance to be used. Therefore, it will not cause the loss of information. Based on the above analysis, we can give the complexity of GFTrust.

**Theorem 1.** *The total time complexity of Algorithms 1 and 2 in GFTrust is* $O(|V||E|^2)$.

**Proof.** For Algorithm 1, each node and edge are considered exactly once. Therefore, it takes the time complexity of $O(|V| + |E|)$. Usually, a network has more edges than nodes, i.e., $|V| < |E|$. Then the complexity can be taken as $O(|E|)$.

For Algorithm 2, the basic operations of augmenting flow and calculating residual capacity (lines 6-11) are in the complexity of $O(|E|)$. Also, the selection of shortest path is the same with breadth-first search, for which the time complexity is $O(|E|)$ in the worst case.

According to the analysis of Observation 3, in the worst case, the iterative number is the same with the Edmonds-Karp algorithm, that is $O(|V||E|)$. Therefore, the total complexity will be $O(|V||E|^2)$. □

Again, according to Observation 3, in most cases, the iterative number is a small constant, for which the total complexity of GFTrust will be $O(|E|)$.

In addition, GFTrust is a local trust metric which is based on a small trusted graph from source $s$ to sink $d$, instead of the whole large OSNs. Local trust metrics scale well to any social network size, as only tiny subsets of relatively constant size are visited (Ziegler and Lausen [17]); we can set the trust threshold and the maximum length of trusted paths, to restrict the size of trusted graphs. In fact, many existing trust evaluation algorithms including Golbeck [2] and Massa and Avesani [27], use the breadth-first search algorithm. From this point of view, our algorithm keeps the same time complexity as others, while solving the two challenges of path dependence and trust decay simultaneously.

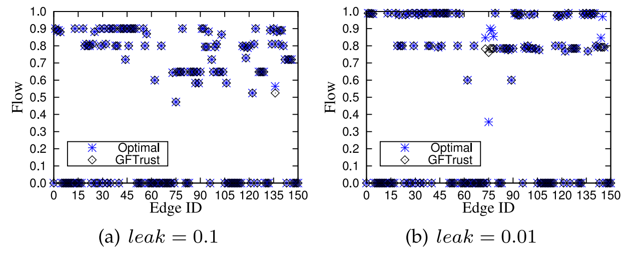


Fig. 7. Comparison of GFTrust and optimal method.

### 6.1.2 Near-Optimal Effect

We check whether GFTrust can gain near-optimal flow, by comparing it with optimal method (implemented by linear programming), in the data set of Kaitiaki (www.kaitiaki.org.nz). Kaitiaki is a small trust network. The data set contains 64 nodes and 178 links. We also assign four levels (i.e., 0.4, 0.6, 0.8, 1.0) of trust for this data set. For each edge in Kaitiaki, we calculate the flow that can pass from its starting node to ending node, through the paths between them (the original edge is masked). Fig. 7 shows that the results of GFTrust are very close to the optimal maximum flow. From this point, the calculated trust of GFTrust is the upper bound estimation, i.e., $s$ can trust $d$ at most in this level.

## 6.2 Basic Desirable Properties

The unique advantages of the GFTrust scheme are that it can deal with both path dependence and trust decay. In addition, GFTrust saves the normalization process that other flow-based methods have to do, and it is more general than non-flow-based models.

### 6.2.1 Ability to Solve Path Dependence

GFTrust can solve the challenge of path dependence without information reuse or loss, which is difficult to avoid in other models. The reasons are as follows: (1) Avoiding information reuse. In GFTrust, the capacity of an edge will be decreased by exactly the amount of flows passing through it. Therefore, the trust value on an edge will not be overused. (2) Avoiding information loss. As mentioned before, in GFTrust, every edge has the chance to be used for sending flows. Therefore, the trust value on every edge is considered.

### 6.2.2 Ability to Solve Trust Decay

GFTrust can deal with trust decay at a fine-grained level. The leakage of each intermediate node can be set flexibly.

### 6.2.3 No Need to Normalize

Since the resulting flow falls in the range of [0,1], it can be taken as a trust value directly. In this sense, GFTrust calculates trust in a summation-like way, which may lead to false positive effects. That is, the resulting trust value may be larger than the direct trust (the ground truth or the expressed trust). However, it makes sense that in real life: if many (more than one) trusted friends recommend someone to us, we will usually take the advice. Moreover, we can eliminate or weaken the false positive effects by increasing the leakage.

#### 6.2.4 Generality

GFTrust is more general than existing trusted graph-based models. For example, other flow-based models can be deemed as a special case of $leak = 0$. Also, MaxT aggregation (which selects the most reliable opinion) in the reliability model can be seen as letting some paths pass zero (0) flows. Furthermore, the flow-based trust model has been verified to be able to deal with multi-dimensional information (Wang and Wu [4]). Thus, GFTrust also bears the advantage.

### 6.3 Special Desirable Properties: Misbehavior Resistance

In OSNs, users may conduct several kinds of misbehavior, such as providing bad service (Josang et al. [30]), Sybil attack (Douceur [31]), bad mouthing (Sun et al. [32]), on-off attack (Sun et al. [32]), conflicting behavior attack (Sun et al. [32]), social spamming (Stringhini et al. [33]). Among them, GFTrust can handle the first two cases, because the unique design has two desirable properties, incentive compatibility (Douceur and Moscibroda [34]) and Sybil tolerance (Viswanath et al. [35]). We consider two types of misbehavior in the interactions between a service provider ($d$) and a customer ($s$, $u$, etc.) in an OSN: (1) $d$ wants to gain more profits but he provides bad services; (2) $d$ tries to false praise himself by making Sybils.

Let $d$ be a service provider, $u$ be a current customer, and $s$ be a potential customer. Let $P(t(s,d))$ be the probability that $s$ will choose $d$'s service, when the trust value from $s$ to $d$ is $t(s,d)$. Let $t(s, v_1, \dots, v_n, d)$ be the trust that is received through path $(s, v_1, \dots, v_n, d)$, and $\Delta t(s,d)$ represents the increment of $t(s,d)$. Let $r$ be the meta-return (e.g. voluntarism, achievement incentive, economic return) of $d$ after he has serviced a customer, and $R = P(t(s,d)) \cdot r$ be the expected return of $d$, from potential customer $s$.

We assume $P(t) \propto t$, i.e., it is proportional to the trust value. We also assume $P(t) >> P(0)$, when $t > 0$. In reality, $P$ may even exponentially increase with $t$. $P(0)$ is the probability that a customer $s$ will randomly select a service provider $d$, without knowing its reputation. We suppose each normal user will provide honest feedback. If $d$ provides good service for $u$, $t(u,d) > 0$; otherwise, $t(u,d) = 0$. In addition, service providers are rational and selfish. They want to get more returns. They will not provide bad service if there are no benefits.

**Property 1 (Social incentive compatibility).** *GFTrust is social incentive compatible in that it can provide nodes with increasing expected value in response to increased contribution.*

**Proof.** Consider the scenario that $d$ serves $u$, and $s$ is a friend of $u$, who thus is a potential customer of $d$. We examine the expected return $R$ of $d$. Since $R = P(t(s,d)) \cdot r$, we focus on how the trust value of $t(s,d)$ will change. The following two cases may happen: (1) $d$ provides good service to $u$; (2) $d$ does not provide good service to $u$.

In case (1), $u$ will give a good reputation $t(u,d)$ on the direct edge to $d$. If $s$ did not have trust in $d$ before, we represent it as $t(s,d) = 0$. Then, the increment of trust will be $\Delta t(s,d) = t(s,u,d)$. Or, if $s$ had known $d$ before, with trust $t(s,d) > 0$. Now, $s$ can revise his trust in $d$ through the new short path $(s,u,d)$. In both conditions, we have $\Delta t(s,d) \geq 0$. In case (2), $u$ will not give a good reputation to $d$. Therefore, $\Delta t(s,d) \leq 0$. Since $P(t)$ is proportional to $t$, the expected return for $d$ is better in case (1). Therefore, the service provider $d$ has the incentive to provide good service. □

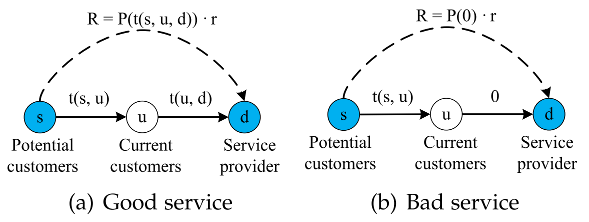


Fig. 8. Expected return of $d$ as a server, $s$ as a potential customer, $u$ as a current customer: (a) $d$ provides good service; (b) $d$ provides bad service.

**Property 2 (Sybil tolerance).** *GFTrust is Sybil tolerant in that a service provider cannot increase its benefits by creating multiple identities to* false praise himself.

**Proof.** A service provider $d$ may create new identities and put them into the network to false praise himself, if this will help to increase his expected return $R$. However, the design of GFTrust makes the Sybil attack unprofitable.

Assume $d$ creates a new identity $d'$, and provides service to $u$ with identity $d'$. Therefore, the new trusted path will be $(s,u,d',d)$ (Fig. 9). When compared with the case without Sybil attack, the newly added trusted path will be $(s,u,d)$ (Fig. 8a). Due to the setting of node leakage, we have $t(s,u,d) \geq t(s,u,d',d)$, then, $P(t(s,u,d)) > P(t(s,u,d',d))$, which will lead to a larger expected return. Therefore, $d$ has no incentive to conduct a Sybil attack. □

It is worth noting that, there is another possible scenario of Sybil attack, where identity $d'$ was inserted at the same distance from $u$. We take it as a different Sybil attack that is beyond our discussion. We can even take $d'$ in this scenario as branch service providers, whose behaviors could be deemed as normal behaviors.

## 7 Experimental Evaluation

We evaluate the performance of GFTrust in two real social network data sets, Epinions and Advogato (the results in Advogato are displayed in Appendix B, available in the online supplemental material). The two data sets are chosen, because they are published data sets that have the ground truth of direct trust values (tru [36]), and they are among the most often-used data sets for evaluating trust prediction performance (e.g., Massa and Avesani [27], Jiang et al. [5], Levien [10], Yao et al. [37]).

The goals of experiments are to (1) compare the effectiveness of GFTrust with other trust evaluation strategies, (2) examine the effects of four leakage functions and other factors, including the maximum path length and the trust threshold, and (3) check in which scenarios GFTrust can provide a correct prediction.

### 7.1 Experimental Design

#### 7.1.1 Evaluation Technique

We use a standard evaluation technique: leave one out (Kohavi [38]). If there is an edge between two nodes (say $s$

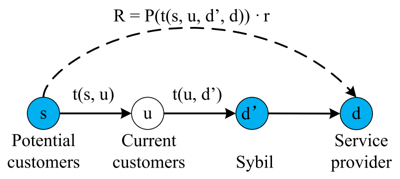


Fig. 9. Expected return of $d$ as a server, $s$ as a potential customer, $u$ as a current customer, and $d$ made a Sybil.

and $d$), that edge is masked. That is to say, the masked value is the ground truth value available from the data set. Next, trust from $s$ to $d$ is calculated through algorithms based on the trusted graph. Then, we compare the calculated value with the masked value.

### 7.1.2 Data Set and Preprocess

We use a real trust network data set of Epinions (www.epinions.com). It is an online community web site where users can write reviews and rate other users' reviews. We use the technique in (Richardson et al. [20]) to transform the trust values in Epinions to be continuous in [0,1], and we use the same subset of Epinions as in (Jiang et al. [5]), which has 3,168 nodes and 51,888 edges.

### 7.1.3 Evaluation Metrics

We consider the metric of *trust prediction accuracy*, which represents the ability to predict whether a user will be trusted or not, and has been commonly used (e.g., Jiang et al. [5], Massa and Avesani [27]).

- Mean Error. $\sum |t^c - t^d|/D$, where $t^c$ is the calculated trust, $t^d$ is the direct trust (i.e., the ground truth or the expressed trust), and $D$ is the total number of edges whose trust can be predicted.
- Precision. $A_t \cap B_t/B_t$, where $A_t$ is the number of edges in which $s$ trusts $d$ directly, and $B_t$ is the number of edges in which $s$ trusts $d$ by the algorithm.
- Recall. $A_t \cap B_t/A_t$.
- FScore. 2·Recall·Precision/(Recall + Precision).

Here, Mean Error is a quantity used to measure how close the predictions are to the direct trust values (i.e., the ground truth or the expressed trust); a smaller Mean Error indicates a higher prediction accuracy. Precision is the fraction of users who are predicted to be trusted, and are really trusted ones (i.e., the ground truth). Recall is the fraction of users who are really trusted, and are subsequently successfully predicted. A higher Precision and Recall indicates a higher prediction accuracy. The FScore metric is used to measure the accuracy using Recall and Precision jointly.

We also consider the metric of *connection coverage*, which is the proportion of edges that have short paths between its pair of nodes (thus their trust can be predicted) in all the edges.

### 7.1.4 Methods for Comparison

In the experiments, we mainly consider the *reliability model* for comparison, since both the reliability model and GFTrust consider the two challenges of path dependence and trust decay.

In the reliability model, the trust value from incoming neighbors ($N_{in}(d)$) to $d$ is the direct trust (i.e., the ground truth or the expressed trust), and the trust value from source $s$ to the nodes in $N_{in}(d)$ is taken as the reliability of the direct trust. Moreover, trust propagation calculates the reliability of a trusted path. Two commonly used methods are Multi and Min. Multi takes the product of trust values in all edges. Min takes the minimum trust value in a path. Trust aggregation among nodes in $N_{in}(d)$ calculates the final trust value. Two commonly used aggregation functions are MaxT and WAveT. MaxT takes the trust value of the most reliable incoming neighbor. WAveT takes the weighted average value of all incoming neighbors. Table 2 shows the equations of trust propagation and aggregation.

TABLE 2
The Basic Operations in the Reliability Model

| Operation | Method | Equation | Condition |
|---|---|---|---|
| Propagation | Multi | $t_p(s, v_n) = \prod t(v_i, v_j)$ | $e(v_i, v_j) \in p$ |
| | Min | $t_p(s, v_n) = \min\{t(v_i, v_j)\}$ | |
| Aggregation | MaxT | $t(s, d) = t(v_j, d), v_j$ *is most reliable* | $v_j \in N_{in}(d)$ |
| | WAveT | $t(s, d) = \sum t(s, v_j) \cdot t(v_j, d)/\sum t(s, v_j)$ | |

We implement five other trust prediction strategies: AveR-MaxT, AveR-WAveT, MaxR-MaxT, MaxR-WAveT, and SWTrust* (Jiang et al. [5]). If there are multiple paths from $s$ to a node in $N_{in}(d)$, AveR will take the average path weight as the reliability, while MaxR will take the maximum one. SWTrust* is an algorithm which makes use of the idea of TidalTrust (Golbeck [2]), and takes the weighted average trust value of all shortest and strongest paths. We set trust threshold $Th \in [0.5, 0.9]$, and max length $L \in [2, 6]$ for experiments.

## 7.2 Experimental Results in Epinions

### 7.2.1 The Effects of Different Strategies

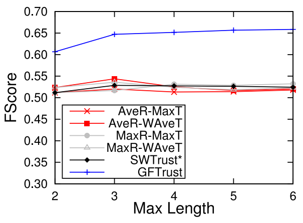
Table 3 and Fig. 10 show the comparison of accuracy in Epinions. It indicates that GFTrust has better performance than other strategies: (1) its FScore is higher, with the improvement being 27.16 percent when $L = 6$; and (2) its Mean Error is lower, with the improvement being 23.72 percent when $L = 6$.

### 7.2.2 The Effects of Leakage Functions

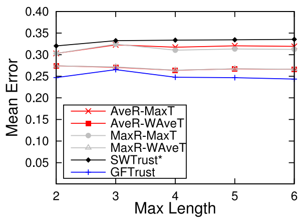
We conduct experiments with the four leakage functions in Fig. 3. Some representative results are presented in Fig. 11. The upper four figures demonstrate the effects of uniform leakage, while the other four figures correspond to non-uniform leakage. We gain some findings:

TABLE 3
Accuracy in Epinions, $L = 4$, $Th = 0.5$, $leakage = 0$

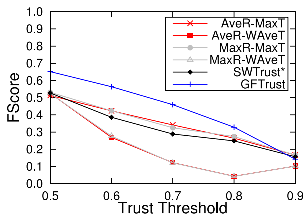
| Method | Mean Error | Recall | Precision | FScore |
|---|---|---|---|---|
| AveR-MaxT | 0.3174 | 0.5385 | 0.4903 | 0.5132 |
| AveR-WAveT | 0.2639 | 0.5342 | 0.5165 | 0.5252 |
| MaxR-MaxT | 0.3105 | 0.5641 | 0.5019 | 0.5312 |
| MaxR-WAveT | 0.2641 | 0.5342 | 0.5144 | 0.5241 |
| SWTrust* | 0.3336 | 0.5556 | 0.5019 | 0.5274 |
| GFTrust | 0.2478 | 0.9872 | 0.4863 | 0.6516 |



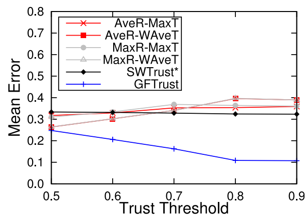
(a) FScore



(b) Mean Error

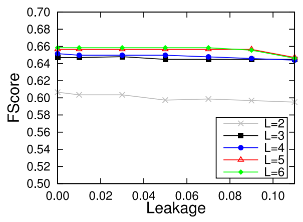


(c) FScore

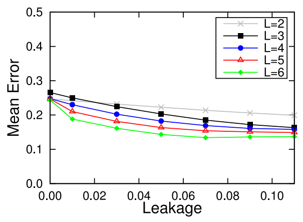


(d) Mean Error

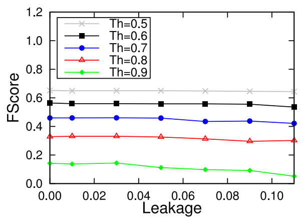
Fig. 10. The accuracy in Epinions.



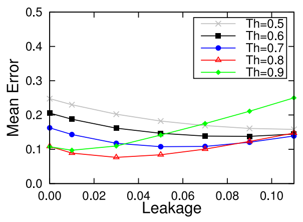
(a) Uniform Leakage



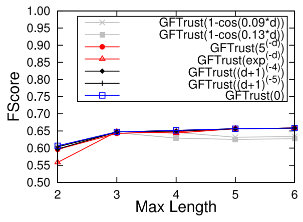
(b) Uniform Leakage



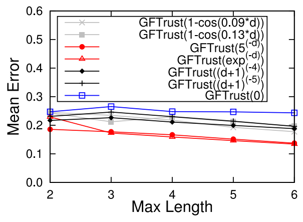
(c) Uniform Leakage



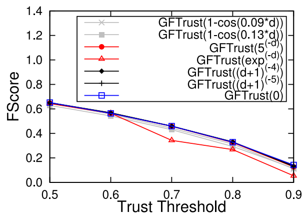
(d) Uniform Leakage



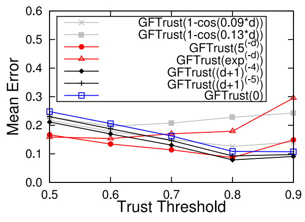
(e) Non-uniform Leakage



(f) Non-uniform Leakage



(g) Non-uniform Leakage



(h) Non-uniform Leakage

Fig. 11. The accuracy of GFTrust with leakage in Epinions.

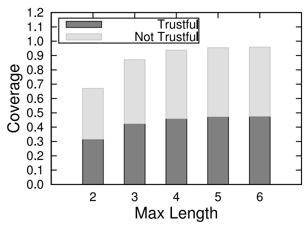
1) The leakage has more positive effects on the Mean Error than on the FScore. E.g., the Mean Error is decreased significantly in Fig. 11b, while the FScore remains relatively stable in Figs. 11a and 11c.
2) The effects are of great difference in different settings of $L$ and $Th$. The changes are sharp with respect to trust threshold $Th$, and much smoother with respect to maximum length $L$. The reason is that the increase of $Th$ takes more of an effect on the available paths, i.e., less paths will be taken as trustful if we set a higher $Th$.
3) When the leakage is large enough, the accuracy (either FScore or Mean Error) begins to be reduced. E.g., when the leakage is larger than 0.06, the FScore is decreased in Fig. 11c, and the Mean Error is increased in Fig. 11d. Therefore, the leakage cannot be too large. In fact, when we set $leak = 0.22$ or even larger, the prediction accuracy is reduced sharply.
4) The effects of non-uniform leakage are even more various with different settings. As shown in Figs. 11e and 11g, the FScore of GFTrust with non-uniform leakage is almost the same with or even lower than GFTrust(0) (i.e., no leakage); meanwhile, Figs. 11f and 11h show that the Mean Error with leakage is lower than that without leakage. In addition, within all the three leakage functions in Eqs. (2), (3), and (4), the last one, polynomial leakage, performs the best.

### *7.2.3 The Effects of Max Length*

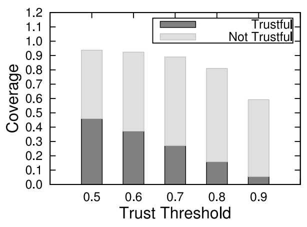
If the max length is large, then there will be more intermediate nodes from $s$ to $d$. Fig. 12a shows that the coverages are increased with increasing $L$, especially when $L$ changes from 2 to 3; the increase is not significant when $L$ changes from 3 to 6. In Epinions, a larger max length leads to little increase of accuracy, as shown in Figs. 10a, 10b, 11a, and 11b. Similar to the coverage, the gap is larger when $L$ changes from 2 to 3 than from 3 to 6. We analyze the reason, and find that there are few paths when $L$ is too small. Although there do exist paths between some pairs of nodes, the number of these paths is fewer.

### *7.2.4 The Effects of Trust Threshold*

Figs. 10c, 10d, 11c, 11d show the accuracy with respect to the trust threshold. During the increase of $Th$ (from 0.5 to 0.9), the FScore decreases sharply, especially when $Th \geq 0.6$. We analyze the reason, and find that, as $Th$ increases, fewer paths will be trusted (due to the decrease of the coverage, as shown in Fig. 12b), which means less evidence can be used to evaluate trust.



(a)



(b)

Fig. 12. The coverage in Epinions.

#### 7.2.5 *Scenarios of Prediction*

Besides testing the above impact factors, we also want to analyze in which conditions GFTrust can make a correct or incorrect decision. We delved into the meta-results, and obtained the following findings. For the pairs of nodes being considered: (1) In the case that GFTrust gives a higher trust than the direct trust, there are usually several short trusted paths between them, the length of which is $L \leq 4$. In fact, only less than three paths are used, which is consistent with Observation 3. (2) On the contrary, in the case that GFTrust gives a lower trust than the direct trust, it usually happens when there are no short, trusted paths. We summarize two such cases: (a) There are several paths, but all the paths are long, and thus, too many intermediate nodes cause too much leakage; (b) there are not enough paths to send all the initial flow.

These findings verify the incentive property of GFTrust. Users (service providers) usually want to get high trust, which can eventually help them get more returns. Then, it is a wise decision to provide good services for more customers. Only in this way can new short and trusted paths be created from their friends (potential customers) to a service provider.

### 7.3 Summary of Experiments

The experimental results validate that GFTrust is effective in improving the trust prediction accuracy, and verify its incentive property. The FScore is higher than that using other methods, and the Mean Error is lower due to the proper setting of the leakage. Among all the four leakage functions, the polynomial leakage performs the best in Epinions.

## 8 Conclusion and Future Work

With the popularity of services and applications in OSNs, the trust issues gain more attention both from service customers and providers. Improving the trust evaluation accuracy will help enhance both the customer experience and the service quality. Due to its high clustering, the path dependence phenomenon is more common in OSNs. Although some trust models have been proposed, the two challenges of path dependence during aggregation and trust decay through propagation have not been well addressed.

Recognizing the similarity between trust propagation and network flow, we design a generalized flow-based trust evaluation scheme, GFTrust. We make use of the nature of flow to deal with trusted path dependence, and model trust decay with node leakage. We analyze the unique advantages of GFTrust, and conduct extensive experiments with real social network data sets. The results validate the effectiveness of GFTrust: the use of flow and the setting of leakage improve the trust prediction accuracy significantly.

GFTrust is the first to address both challenges of path dependence and trust decay, in the domain of trust evaluation in OSNs. It is also the first to introduce the modified generalized flow model into a trust evaluation system. The unique design makes it able to solve the two challenges, save the normalization process, and bear two good properties of social incentive compatibility and Sybil tolerance. In addition, setting the initial trust value as 1 makes GFTrust generous to the newcomers. However, even when we are generous at the beginning, possible Sybil users can only conduct limited malicious behavior. On the other hand, because GFTrust is Sybil-tolerant, it can thus be friendly to newcomers.

Currently, the node leakage functions are designed intuitively. We let the leakage be associated with the distance, and be a type of proportion leakage. In future work, we will improve the design, and also try the approach of fixed value leakage.

## Acknowledgments

This work is supported in part by US National Science Foundation (NSF) grants CNS 149860, CNS 1461932, CNS 1460971, CNS 1439672, CNS 1301774, ECCS 1231461, ECCS 1128209, and CNS 1138963. This work is also supported by NSFC grants 61272151 and 61472451, ISTCP grant 2013DFB10070, the China Hunan Provincial Science & Technology Program under Grant Number 2012GK4106, and the Chinese Fundamental Research Funds for the Central Universities 531107040845. The authors also would like to thank anonymous reviewers who gave valuable suggestions that have helped to improve the quality of the manuscript. Some work of Wenjun Jiang was conducted at Temple University. G. Wang is the Corresponding author.

## References

[1] J. Wu, "Trust mechanisms and their applications in dynamic and mobile computer systems," *Technical Report in TrustCom'09*, Vancouver, Canada, Aug. 2009.

[2] J. Golbeck, "Computing and applying trust in web-based social networks," PhD thesis, Dept. Comput. Sci., College Park, Md, Univ. of Maryland, 2005.

[3] Y. L. Sun, W. Yu, Z. Han, and K. J. R. Liu, "Information theoretic framework of trust modeling and evaluation for ad hoc networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 2, pp. 305–317, Feb. 2006.

[4] G. Wang and J. Wu, "FlowTrust: Trust inference with network flows," *Frontiers Comput. Sci. China*, vol. 5, no. 2, pp. 181–194, 2011.

[5] W. Jiang, G. Wang, and J. Wu, "Generating trusted graphs for trust evaluation in online social networks," *Future Generation Comput. Syst.*, vol. 31, pp. 48–58, 2014.

[6] G. Mahoney, W. Myrvold, and G. C. Shoja, "Generic reliability trust model," *Proc. PST*, 2005, pp. 113–120.

[7] M. Taherian, M. Amini, and R. Jalili, "Trust inference in web-based social networks using resistive networks," in *Proc. 3rd Int. Conf. Internet Web Appl. Serv.*, 2008, pp. 233–238.

[8] D. J. Watts, *Small Worlds: The Dynamics of Networks Between Order and Randomness*. Princeton, NJ, USA: Princeton Univ. Press, 1999.

[9] K. D. Wayne, "Generalized maximum flow algorithms," PhD thesis, School of Operations Research and Information Engineering, Ithaca, NY, Cornell Univ., 1999.

[10] R. Levien and A. Aiken, "Attack resistant trust metrics for public key certification," In *7th USENIX Security Symposium*, pp. 229–242, 1998.

[11] A. Jøsang, R. Hayward, and S. Pope, "Trust network analysis with subjective logic," in *Proc. 29th Aus. Comput. Sci. Conf.*, 2006, pp. 85–94.

[12] C. Lin, N. Cao, S. Liu, S. Papadimitriou, J. Sun, and X. Yan, "Smallblue: Social network analysis for expertise search and collective intelligence," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2009, pp. 1483–1486.

[13] L. Paninski, "Estimation of entropy and mutual information," *Neural Comput.*, vol. 15, no. 6, pp. 1191–1253, Jun. 2003.

[14] T. Bailey, S. Julier, and G. Agamennoni, "On conservative fusion of information with unknown non-gaussian dependence," in *Proc. 5th Int. Conf. Inform. Fusion*, 2012, pp. 1876–1883.

[15] B. Chen and P. K. Varshney, "A Bayesian sampling approach to decision fusion using hierarchical models," *IEEE Trans. Signal Process.*, vol. 50, no. 8, pp. 1809–1818, Aug. 2002.

[16] B. Friedland, *Control System Design: An Introduction to State-Space Methods*. New York, NY, USA: Dover, 2005.

[17] C. N. Ziegler and G. Lausen, "Propagation models for trust and distrust in social networks," *Inf. Syst. Frontiers*, vol. 7(4-5), pp. 337–358, 2005.

[18] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1993.

[19] W. Sherchan, S. Nepal, and C. Paris, "A survey of trust in social networks," *ACM Comput. Surveys*, vol. 45, no. 4, pp. 47:1–47:33, 2013.

[20] M. Richardson, R. Agrawal, and P. Domingos, "Trust management for the semantic web," in *Proc. 2nd Int. Semantic Web Conf.*, 2003, vol. 2870, pp. 351–368.

[21] A. Abdul-Rahman and S. Hailes, "A distributed trust model," in *Proc. Workshop New Security Paradigms*, 1997, pp. 48–60.

[22] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian J. Math.*, vol. 8, pp. 399–404, 1954.

[23] M. A. Nowak, "Five rules for the evolution of cooperation," *Science*, vol. 314(5805), pp. 1560–1563, 2006.

[24] M. L. Manapat, M. A. Nowak, and D. G. Rand, "Information, irrationality, and the evolution of trust," *J. Econ. Behavior Org.*, vol. 90, pp. 57–75, 2012.

[25] D. G. Rand, J. D. Greene, and M. A. Nowak, "Spontaneous giving and calculated greed," *Nature*, vol. 489, no. 7416, pp. 427–430, 2012.

[26] M. A. Perillo and W. B. Heinzelman, "Optimal sensor management under energy and reliability constraints," in *Proc. IEEE Wireless Commun. Netw.*, 2003, vol. 3, pp. 1621–1626.

[27] P. Massa and P. Avesani, "Trust metrics on controversial users: Balancing between tyranny of the majority and echo chambers," *Int. J. Semantic Web Inform. Syst.*, vol. 3, pp. 39–64, 2007.

[28] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *J. Assoc. Comput. Mach.*, vol. 19, no. 2, pp. 248–264, 1972.

[29] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[30] A. Josang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision Support Syst.*, vol. 43, no. 2, pp. 618–644, 2007.

[31] J. R. Douceur, "The Sybil attack," in *Proc. 1st Int. Workshop Peer-to-Peer Syst.*, 2002, pp. 251–260.

[32] Y. L. Sun, Z. Han, W. Yu, and K. J. R. Liu, "A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks," in *Proc. INFOCOM*, 2006, pp. 1–13.

[33] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," in *Proc. 26th Annu. Comput. Security Appl. Conf.*, 2010, pp. 1–9.

[34] J. R. Douceur and T. Moscibroda, "Lottery trees: Motivational deployment of networked systems," in *Proc. Conf. Appl., Technol., Archit. Protocols Comput. Commun.*, 2007, pp. 121–132.

[35] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove, "An analysis of social network-based sybil defenses," in *Proc. Conf. Appl., Technol., Archit. Protocols Comput. Commun.*, 2010, pp. 363–374.

[36] [Online]. Available: http://www.trustlet.org/wiki/Trust_network_datasets, Dec. 2011.

[37] Y. Yao, H. Tong, X. Yan, F. Xu, and J. Lu, "Matri: A multi-aspect and transitive trust inference model," in *Proc. ACM 22nd Int. Conf. World Wide Web*, 2013, pp. 1467–1476.

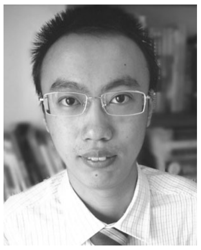
[38] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. 14th Int. Joint Conf. Artif. Intell.*, 1995, pp. 1137–1143.

**Wenjun Jiang** received the bachelor's degree in computer science from Hunan University, P. R. China, in 2004, the Master's degree in computer software and theory from the Huazhong University of Science and Technology, P. R. China, in 2007, and the doctorate degree in computer software and theory from Central South University, P. R. China, in 2014. She has been a visiting PhD student at Temple University for two years. Currently, she is an assistant professor at Hunan University. Her research interests include trust and social influence evaluation models and algorithms in online social networks, recommendation systems, and social network analysis.

**Jie Wu** is the chair and a Laura H. Carnell professor in the Department of Computer and Information Sciences at Temple University. He is also an Intellectual Ventures endowed visiting chair professor at the National Laboratory for Information Science and Technology, Tsinghua University. Prior to joining Temple University, he was a program director at the National Science Foundation and was a distinguished professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. He regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including *IEEE Transactions on Service Computing* and the *Journal of Parallel and Distributed Computing*. He was a general co-chair/chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, and ACM MobiHoc 2014, as well as a program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society distinguished visitor, ACM distinguished speaker, and a chair for the IEEE Technical Committee on Distributed Processing (TCDP). He is a CCF distinguished speaker. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award. He is a fellow of the IEEE.

**Feng Li** received the PhD degree in computer science from Florida Atlantic University in August 2009. His PhD advisor is Prof. Jie Wu. He joined the Department of Computer, Information, and Leadership Technology at Indiana University-Purdue University Indianapolis (IUPUI) as an assistant professor in August 2009. His research interests include the areas of wireless networks and mobile computing, security, and trust management. He has published more than 30 papers in conferences and journals. He is a member of the IEEE.

**Guojun Wang** received the BSc degree in geophysics, MSc degree in computer science, and the PhD degree in computer science from Central South University, China. He is now a chair and professor of the Department of Computer Science and Technology at Central South University. He is also the director of the Trusted Computing Institute of the University. He has been an adjunct professor at Temple University; a visiting scholar at Florida Atlantic University; a visiting researcher at the University of Aizu, Japan; and a research fellow at the Hong Kong Polytechnic University. His research interests include network and information security, Internet of things, and cloud computing. He is a distinguished member of CCF, and a member of the IEEE, ACM, and IEICE.



**Huanyang Zheng** received the BEng in telecommunication engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2012. He is currently working towards the PhD degree in the Department of Computer and Information Sciences, Temple University, Philadelphia, PA. His current research focuses on the delay tolerant networks, social networks, and cloud systems.