# FlowTrust: Trust Inference with Network Flows

Guojun Wang and Jie Wu

**Abstract**—Web-based social networking is increasingly gaining popularity due to the rapid development of computer networking technologies. However, social networking applications still cannot obtain a wider acceptance by many users due to some unresolved issues, such as trust, security, and privacy. Trust, in social networks, mainly studies whether a remote user, called a trustee, behaves as expected by an interested user, called a trustor, through other users, called recommenders. A trusted graph consists of a trustor, a trustee, recommenders, and trust relationships among them. In this paper, we propose a novel FlowTrust approach to model a trusted graph with network flows, and evaluate the maximum amount of trust that can flow among a trusted graph using the network flow theory. FlowTrust supports multi-dimensional trust. We use trust value and confidence level as two trust factors. We deduce four trust metrics from these two trust factors, which are maximum flow of trust value, maximum flow of confidence level, minimum cost of uncertainty with maximum flow of trust value, and minimum cost of untrust with maximum flow of confidence level. We also propose three FlowTrust algorithms to normalize these four trust metrics. We compare our proposed FlowTrust approach with existing RelTrust and CircuitTrust approaches. We have shown that all three approaches are comparable to each other in terms of the inferred trust values. Therefore, FlowTrust is the best of the three since it also supports multi-dimensional trust.

**Index Terms**—Trust inference, multi-dimensional trust, approximate algorithm, network flows, social networks

## 1 Introduction

A social network is a social structure made up of people, called "nodes", which are connected by social relations, such as friendship or business partnership, called "links". Web-based social networking is increasingly gaining popularity due to the rapid development of computer networking technologies. Many social networking websites have been developed, or are emerging, such as MySpace [15], Facebook [3], and LinkedIn [12]. Large social networks are connecting hundreds of millions of people. However, social networking applications still cannot obtain a wider acceptance by many users due to some unresolved issues, such as trust, security, and privacy. We address the trust issue and its models in this paper.


- *G. Wang is with School of Information Science and Engineering, Central South University, Changsha, Hunan Province, P. R. China, 410083. E-mail: csgjwang@mail.csu.edu.cn.*
- *J. Wu is with Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA. E-mail: jiewu@temple.edu.*


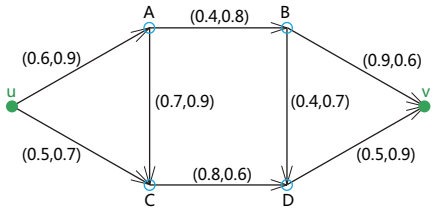


Fig. 1. An example trusted graph.

Trust, in social networks, mainly studies whether a remote user behaves as expected by an interested user through other users. An interested user (called $trustor$) may not be aware of a remote user (called $trustee$) before they make any real communications. That is, direct trust that a trustor puts on a trustee, is often not available in large social networks. Therefore, it is very important to study indirect trust, which a trustor puts on a trustee, through other users (called $recommenders$). This is also called recommendation trust. It should be pointed out that they are different from the referral trust and functional trust proposed in [9] by Jøsang.

Recommendation trust is often evaluated through a trusted graph. A trusted graph consists of a trustor, a trustee, recommenders, and trust relationships among them. A trusted graph is a directed acyclic graph (DAG), where the trustor is the only node that has no edge pointing to itself, and the trustee is the only node that has no edge pointing from itself. Fig. 1 shows an example trusted graph with the trustor $u$ and the trustee $v$. The first number on each edge $(i, j)$ is the total amount of trust value that node $i$ puts on node $j$, and the second number is the total amount of confidence level that $i$ perceives on $j$, regarding the trust value. Both numbers take values from the real interval (0, 1]. The larger these values, the more trustworthiness, or the higher confidence, they have.

Trust inference based on a trusted graph has been studied for many years. However, most existing works used a *graph simplification-based* approach to simplify a trusted graph into node/edge-disjoint multiple paths [16], or a directed series-parallel graph (DSPG) [4]. The *information-loss* problem is caused by the simplification of trusted graphs, and thus some trusted paths and trust relationships cannot be taken into account for trust inference. In order to solve this problem, some researchers proposed to directly deal with any trusted

graph for trust inference, which does not remove any trust relationships from the original trusted graphs. Existing works used a system success diagram in the network reliability theory [2], [17], and a resistive network in the circuit theory [9], to emulate a trusted graph. We call this approach the *graph analogy-based* approach. There are also some approaches using classic mathematical algorithms to infer trust, such as fuzzy logic [10], and Bayesian theory [8].

In this paper, the graph analogy-based approach with the network reliability theory is also called the $RelTrust$ approach [13], and the graph analogy-based approach with the circuit theory is also called the $CircuitTrust$ approach [19]. Their problem is that they can only deal with one trust factor. The reason is that both theoretical frameworks support only one parameter, i.e., reliability and resistance, respectively. The network flow theory has been successfully used to model traffic flows in transportation systems, fluids in pipes, and currents in electrical circuits. The trust metrics in trusted graphs have common attributes similar to electrical currents and transportation traffics. In this paper, we made the first attempt at measuring trust using the network flow theory to achieve the trust evaluation in trust management systems.

Recent research shows that trust is inherently multi-dimensional [5], [7], [23], [21], including multiple trust factors, such as trust value and confidence level. The graph simplification-based approach can support multi-dimensional trust. But, it has the information-loss problem. The graph analogy-based approach uses all the information in a trusted graph. But, it cannot support multi-dimensional trust. Our work leverages the graph analogy-based approach and considers the multi-dimensional trust as well. Our contributions are three-fold:

1) Traditionally, network flows are used to model traffics in transportation systems, fluids in pipes, or currents in electrical circuits. We propose a novel FlowTrust approach to model a trusted graph with network flows, and evaluate the maximum amount of trust that can flow among a trusted graph using the network flow theory.
2) FlowTrust supports multi-dimensional trust. We use trust value and confidence level as two trust factors. We deduce four trust metrics from these two trust factors, which are maximum flow of trust value, maximum flow of confidence level, minimum cost of uncertainty with maximum flow of trust value, and minimum cost of untrust with maximum flow of confidence level. We also propose three FlowTrust algorithms to normalize these four trust metrics.
3) We compare our proposed FlowTrust approach with existing RelTrust and CircuitTrust approaches. Since RelTrust and CircuitTrust do not support multi-dimensional trust, we only compare the inferred trust values by all three approaches. We have shown that all three approaches are comparable to each other in terms of the inferred trust values. Therefore, FlowTrust is the best of the three since it also supports multi-dimensional trust.

The rest of this paper is organized as follows: In the next section, we introduce some related works. In Section 3, we overview the proposed FlowTrust approach with definitions of maximum flow and minimum cost flow, and four trust metrics based on them. In Section 4, we propose three algorithms in FlowTrust. In Section 5, we conduct simulation studies on the FlowTrust approach in comparison with the RelTrust and CircuitTrust approaches. Finally, in Section 6, we conclude this paper and shed some light on future works.

## 2 Related Works

Trust inference based on trusted graphs has been studied for many years. Our proposed FlowTrust is related to both the graph simplification-based approach and the graph analogy-based approach by taking the advantages of both approaches, while limiting their disadvantages.

### 2.1 Graph Simplification-based Approach

Mui et al [14], Theodorakopoulos et al [20], Sun et al [18], and Golbeck et al [6] proposed to use node/edge-disjoint multiple paths between two unknown participants, selected from a trusted graph. In the case of node-disjointness, each node appears in at most one path, that is, each node, except for the trustor itself, makes at most one recommendation in a trusted graph. In the case of edge-disjointness, each edge appears in at most one path, that is, a recommendation from one node to another node can be used at most once among those multiple paths. The graph simplification-based approach with disjoint paths has the advantage of achieving fairness since it restricts the impact of each node or each edge to the minimum.

Zuo et al [24] studied a framework for trust evaluation based on a set of trusted chains in a trusted graph. In order to maximize the trust value of the trustee evaluated based on a trusted graph, they proposed a notion of *a base trusted chains set*. This notion is similar, but different from the notion of node/edge-disjoint multiple paths because the inferred trust value will be maximized when this set is used. The problem is that there may not exist an efficient algorithm to identify the set. So, the authors proposed an algorithm to identify the set through exhaustive enumeration.

In [11], Jøsang et al presented a method for simplifying a complex network to a directed series-parallel graph (DSPG), which can be constructed by a sequence of series and parallel compositions. A DSPG is more general than node/edge-disjoint multiple paths. However, a DSPG may not be able to achieve fairness because some nodes or edges may be shared by multiple paths.

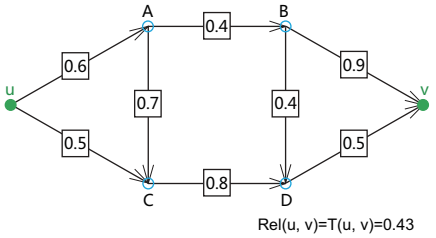


Fig. 2. An example network of failure-prone elements transformed from the example trusted graph in Fig. 1.

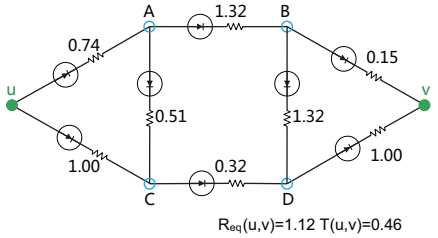


Fig. 3. An example resistive network transformed from the example trusted graph in Fig. 1 (Base $b = 2$).

### 2.2 Graph Analogy-based Approach

Since the graph simplification-based approach has the information-loss problem, researchers are exploring the similarity between trusted graphs in the trust domain, and networks or diagrams in other domains. Once such a similarity is identified, a trusted graph can be emulated by a network or diagram in other domains. Then, the trust value can be inferred by existing methods in other domains. So, we call this approach the *graph analogy-based* approach, including RelTrust [13] and CircuitTrust [19].

In RelTrust, the basic idea is to emulate a trusted graph from a trustor $u$ to a trustee $v$ with a network of failure-prone elements between two terminals $u$ and $v$, and then apply the network reliability theory to calculate the probability that the network is operational, regarding $u$ and $v$. Finally, this probability, denoted $Rel(u, v)$, is viewed as the inferred trust value that the trusted graph has, denoted $T(u, v)$. That is:

$$Rel(u, v) = T(u, v).$$

Fig. 2 shows an example network of failure-prone elements, which is transformed from the example trusted graph in Fig. 1. Notice that only the trust values, i.e., the first numbers at each pair of numbers associated with each edge in the example trusted graph, are used for making the graph transformation.

In CircuitTrust, the basic idea is to emulate a trusted graph with a resistive network, using a logarithmic function to map between the trust values and the resistance values, that is:

$$r = -\log_b t,$$

where $r$ is the resistance value of the resistor assigned to the trust relation, which has the value $t$, and $b(> 0)$ is the base of the logarithmic function. Then, we compute the equivalent resistance value between $u$ and $v$, denoted $R_{eq}(u, v)$. Finally, the inferred trust value between $u$ and $v$, called $T(u, v)$, can be calculated from the following equation:

$$T(u, v) = b^{-R_{eq}(u,v)}.$$

Fig. 3 shows an example resistive network, which is transformed from the example trusted graph in Fig. 1. Notice that only the trust values, i.e., the first numbers at each pair of numbers associated with each edge in the example trusted graph, are used for making the graph transformation.

The problem with both RelTrust and CircuitTrust is that they can only deal with one trust factor, i.e., the overall trust value or trustworthiness. In this paper, we propose to deal with multi-dimensional trust, i.e., multiple trust factors, using a novel graph analogy-based approach, called FlowTrust.

## 3 Preliminaries

Let $G = (V, E)$ be a DAG, representing a trusted graph, where $V$ is the set of nodes in $G$, and $E$ is the set of edges in $G$. Let $u$ and $v$ be two distinct nodes in $G$, where $u$ stands for a trustor and $v$ stands for a trustee. Each edge in $E$, for example, from node $i$ to node $j$, has two trust factors, namely, trust value $t(i, j)$ and confidence level $c(i, j)$, both of which take values from the real interval (0, 1].

In terms of the network flow theory, the two trust factors are also called *capacities*. Here, a capacity represents the maximum amount of flow that can pass through an edge. For example, in the case of the capacity of trust value $t(i, j)$, it represents the maximum amount of flow of trust value that can pass through the edge from node $i$ to node $j$. In this section, we first introduce the network flow theory with the maximum flow and minimum cost flow problems. Then, we define four trust metrics by applying the network flow theory.

Let $G = (V, E)$ be a DAG with $u$ and $v$ being the source and the sink of $V$, respectively. The capacity of an edge from node $i$ to node $j$ is a mapping $cap : E \to R^+$, denoted $cap(i, j)$, which is a non-negative number, representing the maximum amount of commodity that can "flow" through the edge per unit of time in a steady-state situation. A flow from node $i$ to node $j$ is a mapping $f : E \to R^+$, denoted $f(i, j)$, subject to the following two constraints:

1) $f(i, j) \leq cap(i, j)$, for each $(i, j) \in E$ (capacity constraint);
2) $\sum_{i:(i,j)\in E} f(i, j) = \sum_{i:(j,i)\in E} f(j, i)$, for each $j \in V \setminus \{u, v\}$ (conservation law).

The flow of a network $G$ is defined by:

$$F = \sum_{i:(u,i)\in E} f(u, i),$$

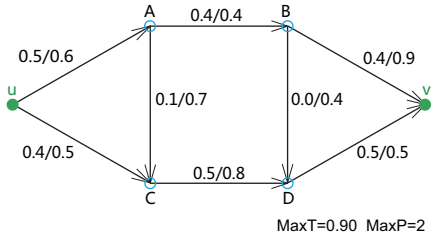


Fig. 4. An example maximum flow of trust value.

where $u$ is the source of $G$. Equivalently, it can also be:

$$F = \sum_{i:(i,v)\in E} f(i,v),$$

where $v$ is the sink of $G$. It represents the amount of flow passing from the source to the sink. The maximum flow problem is to maximize $F$, that is, to route as much flow as possible from $u$ to $v$.

Then, we introduce the minimum cost flow problem. Let the cost of an edge ($cost(i,j)$), which is a non-negative number, represents the amount of "cost" when a unit of commodity "flows" through the edge. The minimum cost flow problem is to minimize the total cost:

$$\sum_{i,j\in V} cost(i,j) * f(i,j)$$

under the constraint of the same network flow $F$. A variant of this problem is to minimize the total cost of the flow and to maximize the network flow at the same time, which is also called the minimum cost maximum flow problem.

Just like the concept of "opinion" to describe the trust metrics based on subjective logic theory, we define four trust metrics with trust factors based on the network flow theory in the following:

*Definition 1: Maximum Flow of Trust Value ($MaxT$).* Given $G = (V, E)$, a trustor $u$, and a trustee $v$ in $V$, when the capacity of an edge $cap(i,j)$ is given by the trust value $t(i,j)$ that node $i$ puts on node $j$, the maximum flow of $G$ is called the maximum flow of trust value.

Fig. 4 shows an example maximum flow of trust value. The second number on each edge $(i,j)$ is the capacity (here, the total amount of trust value) that node $i$ puts on node $j$, and the first number is the flow (here, the portion of trust value) that node $i$ takes out of the total amount of trust value on node $j$. This figure shows that the maximum flow of trust value is 0.9, which is $0.5 + 0.4 = 0.9$ from the perspective of $u$, and $0.4 + 0.5 = 0.9$ from the perspective of $v$. In addition, $MaxP$, shown in this figure, is the maximum number of edge-disjoint paths from $u$ to $v$, which will be further discussed in the next section to normalize the trust metrics. For example, the normalized trust value in this example will be:

$$\frac{MaxT}{Maxp} = \frac{0.9}{2} = 0.45.$$

*Definition 2: Maximum Flow of Confidence Level ($MaxC$).* Given $G = (V, E)$, a trustor $u$, and a trustee $v$ in $V$, when the capacity of an edge $cap(i,j)$ is given by the confidence level $c(i,j)$, regarding the trust value that node $i$ puts on node $j$, the maximum flow of $G$ is called the maximum flow of confidence level.

Due to space restriction, we cannot draw a figure for an example maximum flow of confidence level, which would be similar to Fig. 4.

*Definition 3: Minimum Cost of Uncertainty with Maximum Flow of Trust Value ($Min\overline{C}MaxT$).* Given $G = (V, E)$, a trustor $u$, and a trustee $v$ in $V$. Also, given the capacity of an edge $cap(i,j)$, represented by the trust value $t(i,j)$ that node $i$ puts on node $j$, and the cost of an edge $cost(i,j)$, represented by the uncertainty that is simply defined as "1-confidence value", i.e., $1 - c(i,j)$. Then, the minimum cost maximum flow problem is defined as minimizing the uncertainty and maximizing the flow of the trust value at the same time.

Generally speaking, we can find a maximum of the trust value with the "highest" total confidence. Fig. 5 shows an example minimum cost of uncertainty with maximum flow of trust value. The same goes for Fig. 4, where it shows the same amount of maximum flow of trust value ($MaxT = 0.9$, but this figure also shows the minimum cost of uncertainty ($Min\overline{C}MaxT$=0.66).

*Definition 4: Minimum Cost of Untrust with Maximum Flow of Confidence Level ($Min\overline{T}MaxC$).* Given $G = (V, E)$, a trustor $u$, and a trustee $v$ in $V$. Also, given the capacity of an edge $cap(i,j)$, represented by the confidence level $c(i,j)$, regarding the trust value that node $i$ puts on node $j$, and the cost of an edge $cost(i,j)$, represented by the untrust that is simply defined as "1-trust value", i.e., $1 - t(i,j)$. Then, the minimum cost maximum flow problem is defined as minimizing the untrust and maximizing the flow of the confidence level at the same time.

Generally speaking, we find a maximum of the confidence level with the "highest" total trust. Due to space restriction, however, we cannot draw a figure for an example minimum cost of untrust and maximum flow of confidence level, which would be similar to Fig. 5.

## 4 FlowTrust Algorithms

FlowTrust contains three algorithms, each of which takes a trusted graph, and also the trust value and confidence level factors for each edge in the graph, as inputs, and outputs four normalized trust metrics to show how much trust the trusted graph contains. We name the three algorithms "Basic FlowTrust algorithm" (B-FlowTrust), "Edge preprocessing-based FlowTrust algorithm" (E-FlowTrust), and "Graph simplification-based FlowTrust algorithm" (G-FlowTrust), respectively. The proposed three algorithms do not rely on each other, but they are still comparable to a certain degree. All algorithms are based on the network flow theory and the maximum flow and minimum cost flow, in particular.

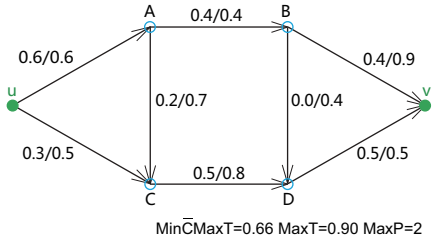


Fig. 5. An example minimum cost of uncertainty with maximum flow of trust value.

### 4.1 Algorithm Design

#### 4.1.1 Basic FlowTrust Algorithm (B-FlowTrust)

Definitions 1-4 implicitly show how to compute those four trust metrics of our FlowTrust approach by applying the maximum flow and minimum cost flow algorithms. However, those metrics have strong relationships with the sizes of the trusted graphs, and particularly, the maximum numbers of edge-disjoint paths in the trusted graphs. Generally speaking, if the maximum number of edge-disjoint paths of a trusted graph is larger, all four of those metrics also become larger. Therefore, we propose to normalize those metrics by dividing them with the maximum number of edge-disjoint paths in the trusted graph. That is, our basic FlowTrust algorithm (B-FlowTrust) first computes those four trust metrics, and then deduces four normalized trust metrics, accordingly. Below, we only show this basic algorithm from the perspective of *Maximum Flow of Trust Value* ($MaxT$) (See Definition 1).

Given a trusted graph $G = (V, E)$, and two distinct nodes $u$ and $v$ in $V$, we first find the maximum flow of trust value from $u$ to $v$, denoted $MaxT$, and then we find the maximum number of edge-disjoint paths from $u$ to $v$, denoted $MaxP$. Finally, we divide $MaxT$ by $MaxP$, denoted $T_b$, representing the normalized trust value inferred from this basic algorithm. If we view the $MaxP$ paths as a "backbone" of the trusted graph $G$, then $T_b$ can be viewed as the average flow of trust value that "flows" through each of the $MaxP$ paths. Algorithm 1 shows this basic algorithm. Notice that we get $MaxP$ in the same way as $MaxT$, by applying the maximum flow algorithm, but the edge capacity (here, the trust value) of each edge in the trusted graph is set to 1.

**Algorithm 1** Basic FlowTrust Algorithm from the perspective of *Maximum Flow of Trust Value* ($MaxT$)

**Input**: A trusted graph $G = (V, E)$, $u, v \in V$,
  and the trust value $t(i,j), \forall (i,j) \in E$
**Output**: The normalized trust value $T_b$
Let $MaxT$ be the maximum flow of trust value of $G$
  from $u$ to $v$ with $t(i,j)$ as the edge capacity
Let $MaxP$ be the maximum number of edge-disjoint
  paths of $G$ from $u$ to $v$
Return $T_b = \frac{MaxT}{MaxP}$

#### 4.1.2 Edge Preprocessing-based FlowTrust Algorithm (E-FlowTrust)

The time complexities of the maximum flow and minimum cost flow algorithms are relatively high. Therefore, some approximate and/or distributed algorithms have been developed. In this subsection, we propose an approximate approach to normalizing those trust metrics, as defined in Section 3 (See Definitions 1-4). In this approach, each trust metric is automatically normalized without the need to compute the maximum number of edge-disjoint paths in the trusted graph. The basic idea is to preprocess the trust value or confidence level on each edge before applying the maximum flow and minimum cost flow algorithms.

**Algorithm 2** Edge Preprocessing-based FlowTrust Algorithm from the perspective of *Maximum Flow of Trust Value* ($MaxT$)

**Input**: A trusted graph $G = (V, E)$, $u, v \in V$,
  and the trust value $t(i,j), \forall (i,j) \in E$
**Output**: The normalized trust value $T_e$
**For** each $t(i,j)$ **Do**{
  Let $\delta_i$ be node degree of $i$ (indegree + outdegree)
  Let $\delta_j$ be node degree of $j$ (indegree + outdegree)
  **If** $\delta_i \geq \delta_j$ **Then** $t'(i,j) = \frac{t(i,j)}{\delta_i}$
  **Else** $t'(i,j) = \frac{t(i,j)}{\delta_j}$
}
Let $MaxT$ be the maximum flow of trust value of $G$
  from $u$ to $v$ with $t'(i,j)$ as the edge capacity
Return $T_e = MaxT$

Similar to B-FlowTrust, the E-FlowTrust algorithm also normalizes those four trust metrics in Definitions 1-4. Below, we only show this approximate algorithm from the perspective of *Maximum Flow of Trust Value* ($MaxT$) (See Definition 1).

#### 4.1.3 Graph Simplification-based FlowTrust Algorithm (G-FlowTrust)

Most existing works on trust inference are based on graph simplification. Node/edge-disjoint paths are most often used for simplifying the trusted graphs. In our FlowTrust approach, edge-disjoint paths are of significant importance since the maximum number of edge-disjoint paths from the trustor $u$ to the trustee $v$ in a trusted graph can be considered a "backbone" of the trusted graph, as discussed in the subsection of the basic FlowTrust algorithm (B-FlowTrust). In this subsection, we first introduce the graph simplification-based FlowTrust algorithm (G-FlowTrust), and then we show the relationship between the normalized trust metrics under B-FlowTrust, and those under G-FlowTrust.

**Algorithm 3** Graph Simplification-based FlowTrust Algorithm from the perspective of *Maximum Flow of Trust Value ($MaxT$)*

**Input**: A trusted graph $G = (V, E)$, $u, v \in V$,
  and the trust value $t(i, j), \forall (i, j) \in E$
**Output**: The normalized trust value $T_g$
Let $MaxP$ be the maximum number of edge-disjoint
  paths from $u$ to $v$ in $G$
Let $TotalP$ be the total number of simple paths
  from $u$ to $v$ in $G$
Find the combination of $MaxP$ out of $TotalP$ paths
  such that the following conditions satisfy:
  (1) The $MaxP$ paths are edge-disjoint;
  (2) $MaxT$ is the maximum flow of trust value of
    the simplified graph with only the $MaxP$ paths;
  (3) $MaxT$ is maximum among all combinations of
    the edge-disjoint $MaxP$ out of $TotalP$ paths.
Return $T_g = \frac{MaxT}{MaxP}$

## 4.2 Algorithm Analysis

### 4.2.1 Analysis of B-FlowTrust

We prove that $T_b$, inferred from the proposed basic algorithm (B-FlowTrust), falls between the real interval [0, 1]. If $T_b$ is larger, e.g., close to 1, then the trustor $u$ intends to put more trust on the trustee $v$, based on the trusted graph. In contrast, if $T_b$ is very small, e.g., close to 0, then $u$ may not trust $v$, and thus $u$ may not communicate with $v$, based on this trust inference.

*Theorem 1:* The normalized trust value $T_b$, inferred from B-FlowTrust, falls between the real interval [0, 1].

*Proof*: In a trusted graph $G = (V, E)$, from truster $u$ to trustee $v$, the edge capacity (here, the trust value) of each edge, $t(i, j), \forall (i, j) \in E$, has a value from the real interval (0, 1]. $MaxT$ is computed by applying the maximum flow algorithm on $G$ from $u$ to $v$, with $t(i, j)$ as the edge capacity. Then, by setting the edge capacity $t'(i, j)$ to be 1 for all edges in $E$, $MaxP$ can be computed by applying the same maximum flow algorithm on $G$ from $u$ to $v$. It is obvious that $0 \leq MaxT \leq MaxP$ since $t(i, j) \leq t'(i, j), \forall (i, j) \in E$. We assume that there is at least one path from the trustor $u$ to the trustee $v$ in the trusted graph $G$, that is, $1 \leq MaxP$. So:

$$0 \leq T_b = \frac{MaxT}{MaxP} \leq 1.$$

This proves the theorem. ■

Similarly, from the perspective of *Maximum Flow of Confidence Level ($MaxC$)* (See Definition 2), we can obtain the normalized confidence level, denoted $C_b$=$\frac{MaxC}{MaxP}$, which also falls between the real interval [0, 1].

Also, from the perspective of *Minimum Cost Uncertainty with Maximum Flow of Trust Value ($Min\overline{C}MaxT$)* (See Definition 3), we can obtain the normalized metric, denoted $\overline{C}T_b$=$\frac{Min\overline{C}MaxT}{MaxP}$. This normalized metric can be considered the "average" value by dividing the "total" minimum cost flow with the "maximum" number of edge-disjoint paths in the trusted graph. Notice that this normalized metric is not necessarily a value that falls between the real interval [0, 1].

Similar to $\overline{C}T_b$, from the perspective of *Minimum Cost Untrust with Maximum Flow of Confidence Level ($Min\overline{T}MaxC$)* (See Definition 4), we can also obtain the normalized metric, denoted $\overline{T}C_b$=$\frac{Min\overline{T}MaxC}{MaxP}$.

### 4.2.2 Analysis of E-FlowTrust

We prove that $T_e$, inferred from the edge preprocessing-based FlowTrust algorithm (E-FlowTrust), falls between the real interval [0, 1]. We also show that it is an approximation of $T_b$, inferred from B-FlowTrust, by showing a special type of trusted graphs, where $T_e$=$T_b$.

*Theorem 2:* The normalized trust value $T_e$, inferred from E-FlowTrust, falls between the real interval [0, 1].

*Proof*: Consider the trustor $u$ in the trusted graph $G$. Let $\delta_u^{in}$ be the indegree of $u$, and $\delta_u^{out}$ be the outdegree of $u$. $\delta_u^{in} = 0$ since $u$ is the source of the flow network, and $\delta_u^{out} > 0$ since we consider there is at least one path between the source and the sink (that is, the trustee $v$) of the flow network. So, $\delta_u = \delta_u^{out}$. Let $u_k (0 \leq k \leq \delta_u - 1)$ be the $k$th associated node corresponding to the $k$th outgoing edge of $u$. Let $t'(u, u_k)$ be the capacity (trust value) after the original $t(u, u_k)$ has been preprocessed by the E-FlowTrust algorithm, where $0 \leq t(u, u_k) \leq 1$. Let $t''(u, u_k)$=$\frac{t(u,u_k)}{\delta_u}$. It is easy to see that:

$$0 \leq \sum_{k=0}^{\delta_u} t'(u, u_k) \leq \sum_{k=0}^{\delta_u} t''(u, u_k) \leq 1$$

because $t'(u, u_k) \leq t''(u, u_k)$. This shows that the total capacity of all the outgoing edges of $u$ is not greater than 1. This implies that the normalized trust value $T_e$ after edge preprocessing, is also not greater than 1. ■

Then, we show that $T_e$, inferred from this approximate algorithm, is equal to $T_b$, inferred from the basic algorithm for a special type of trusted graphs. Below, is the theorem and its proof.

*Theorem 3:* Given $G = (V, E)$, the trustor $u$ and the trustee $v$ in $G$. If all the nodes in $G$ have the same node degree (including indegree and outdegree), and if this node degree is equal to the maximum number of edge-disjoint paths from the trustor $u$ to the trustee $v$, then $T_e$, inferred from E-FlowTrust, is equal to $T_b$, inferred from B-FlowTrust.

*Proof*: Let $MaxT$ be the maximum flow of trust value of $G = (V, E)$ from $u$ to $v$ with $t(i, j)$ as the edge capacity for $(i, j) \in E$. Let $MaxP$ be the maximum number of edge-disjoint paths in $G$ from $u$ to $v$. Then, $T_b = \frac{MaxT}{MaxP}$. Let the node degree for each node in the graph be $\delta (\geq 1)$. So, $t'(i, j) = \frac{t(i,j)}{\delta}$. Let $MaxT'$ be the maximum flow of $G = (V, E)$ from $u$ to $v$ with $t'(i, j)$ as the edge capacity for $(i, j) \in E$. It is easy to see that:

$$T_e = MaxT' = \frac{MaxT}{\delta} = \frac{MaxT}{MaxP} = T_b.$$

This proves the theorem. ■

This theorem shows a salient feature of E-FlowTrust: If a trusted graph has a similar (if not the same) node degree for all the nodes, then the normalized trust value $T_e$, inferred from the approximate algorithm, will be close to the normalized trust value $T_b$, inferred from the basic algorithm. With more similarity between the trusted graph in question, and such a special type of trusted graphs assumed in this theorem, the approximate algorithm will infer the normalized trust value more closely to that of the basic algorithm.

Up until now, we have shown only the normalized trust value $T_e$ under E-FlowTrust. Similarly, we can also obtain the normalized *confidence level* $C_e$, the normalized *minimum cost uncertainty with maximum flow of trust value* ($\overline{C}T_e$), and the normalized *minimum cost untrust with maximum flow of confidence level* ($\overline{T}C_e$). Due to space restriction, we will not show them in detail. However, we point out that, when computing $\overline{C}T_e$, only the trust value $t(i,j)$ for all edges will be preprocessed, while the confidence level $c(i,j)$ for all edges will not be preprocessed. Also, when computing $\overline{T}C_e$, we only preprocess the confidence level $c(i,j)$ for all edges, and the trust value $t(i,j)$ for all edges will not be changed.

#### 4.2.3 Analysis of G-FlowTrust

The time complexity of this algorithm is rather high because it is required to enumerate all combinations of the edge-disjoint $MaxP$ out of $TotalP$ paths. The total number of such combinations is:

$$C_{TotalP}^{MaxP} = \frac{TotalP!}{MaxP!*(TotalP-MaxP)!}.$$

For each combination, it is required to compute $MaxT$, which is the maximum flow of trust value of the simplified graph with only the $MaxP$ paths. So, it is problematic in terms of the time complexity. Fortunately, we aren't too concerned about the time complexity of this algorithm because we only need to show that B-FlowTrust and E-FlowTrust are better than this algorithm in terms of the trust metrics inferred by these algorithms. Below, Theorem 4 shows that the normalized trust value, inferred from G-FlowTrust, is less than or equal to that inferred from B-FlowTrust. This is reasonable since G-FlowTrust used a simplified trusted graph, and thus some trust relationships will be neglected in the trust inference. In contrast, B-FlowTrust takes into account all the trust relationships in the whole trusted graph, and thus the trustor $u$ can get the most out of the trusted graph regarding the trustworthiness on the trustee $v$.

*Theorem 4:* The normalized trust value $T_g$, inferred from G-FlowTrust, is less than or equal to the normalized trust value $T_b$, inferred from B-FlowTrust.

*Proof*: Let $MaxP$ be the maximum number of edge-disjoint paths in $G$ from the trustor $u$ to the trustee $v$. By exhaustive enumeration, there are a total number of $TotalP$ simple paths in $G$. Apparently, any $MaxP$ edge-disjoint paths out of the $TotalP$ paths in $G$, denoted $S$, is a sub-graph of $G$. That is, $S \subseteq G$. Since $T_g$ is inferred from $S$ (in fact, the sub-graph from which the maximum normalized trust value is inferred), and $T_b$ is inferred from $G$, thus $T_g \leq T_b$. ■

Up until now, we have only shown G-FlowTrust from the perspective of *Maximum Flow of Trust Value* ($MaxT$), i.e., the normalized *trust value* $T_g$. For brevity, we will not introduce, in detail, how to infer the normalized *confidence level* $C_g$, the normalized *minimum cost uncertainty with maximum flow of trust value* ($\overline{C}T_g$), and the normalized *minimum cost untrust with maximum flow of confidence level* ($\overline{T}C_g$).

From the above analysis, we can see that all three algorithms can deduce normalized trust metrics. B-FlowTrust uses intact trust value and confidence level factors as inputs, and it can deduce normalized trust metrics, but it does not have a good performance if the trusted graph is too large, and thus it is not suitable for large-scale trusted graphs. E-FlowTrust is an approximate algorithm. If a trusted graph has a similar node degree for all the nodes, then the normalized trust value inferred from E-FlowTrust will be close to the value inferred from B-FlowTrust. G-FlowTrust is an approximation of B-FlowTrust based on graph simplification, but its time complexity is rather high because graph simplification is very time consuming. Notice that the deduced trust value by G-FlowTrust is less than or equal to that inferred from B-FlowTrust.

## 5 Simulation Studies

In this section, we compare our proposed FlowTrust approach with existing RelTrust and CircuitTrust approaches by extensive simulations. In RelTrust, we use the *factoring* mechanism [2] (also called the *short/open* mechanism [17]) to calculate the reliability between two terminals in a network of failure-prone elements. In CircuitTrust, we use the *nodal analysis* [9] to calculate the equivalent resistance between two points in a resistive network. Also, we compare our three proposed FlowTrust algorithms. For simplicity, we suppose that trust values and confidence levels have been obtained from mutual interaction experience in real-world systems, such as web-based social networks, P2P file-sharing systems, and e-commence systems. So, we can directly use such values in these three algorithms.

### 5.1 Simulation Scenarios

1) Scenario I (Example trusted graphs): We present the inferred trust values, based on example trusted graphs, using our FlowTrust in comparison with existing RelTrust and CircuitTrust. These graphs are also used to infer the normalized trust metrics by our three FlowTrust algorithms.
2) Scenario II (Random trusted graphs): We randomly generate many trusted graphs for statistical analysis on the inferred trust values using RelTrust, CircuitTrust, and FlowTrust approaches. We also

make a statistical analysis on the normalized trust metrics using our three FlowTrust algorithms.

### 5.2 Simulation Parameters

Simulation parameters for Scenario I, i.e., example trusted graphs, are shown in Table 1. Notice that a threshold of trust value ($T-THR$) means that all the trust values are randomly chosen from the real interval [*T-THR*, 1]. We present "LOW" threshold (*T-THR*=0.5) and "HIGH" threshold (*T=THR*)=0.8, for each combination of the number of nodes (#*N*) and the number of edges (#*E*). For brevity, other normalized trust metrics, such as normalized confidence level, are not shown in this table. In our simulations, it is required that there are at least two edge-disjoint trusted paths between a trustor and a trustee, when generating such example trusted graphs.

TABLE 1
Simulation parameters for Scenario I

| Parameter | Description | Value |
|---|---|---|
| #N | Number of nodes | 4-20 |
| #E | Number of edges | 5-30 |
| T-THR | Threshold of trust value | 0.5, 0.8 |
| C-THR | Threshold of confidence level | 0.5, 0.8 |
| MaxP | Maximum number of edge-disjoint paths | $\geq$2 |
| $\mathrm{T_{Rel}}$ | Inferred trust value using RelTrust | [0, 1] |
| $\mathrm{T_{Cir}}$ | Inferred trust value using CircuitTrust | [0, 1] |
| $\mathrm{T_b}$ | Normalized trust value using B-FlowTrust | [0, 1] |
| $\mathrm{T_e}$ | Normalized trust value using E-FlowTrust | [0, 1] |
| $\mathrm{T_g}$ | Normalized trust value using G-FlowTrust | [0, 1] |

Most simulation parameters for Scenario II are the same for Scenario I, as shown in Table 1. Due to space restriction, we only show the different simulation parameters for Scenario II in Table 2. For each combination of the number of nodes (#*N*) and the number of edges (#*E*), we randomly generate many simulation runs (#*RUNS*). We calculate the average differences between different approaches and different algorithms. On the one hand, in order to compare the RelTrust, CircuitTrust, and FlowTrust approaches, we select the inferred trust values by RelTrust as a baseline to calculate the absolute differences between other approaches and RelTrust, and finally, the arithmetic average of these absolute differences. On the other hand, in order to compare the B-FlowTrust, E-FlowTrust, and G-FlowTrust algorithms, we select the normalized trust metrics by B-FlowTrust as a baseline to calculate the absolute differences between other algorithms and B-FlowTrust, and finally, the arithmetic average of these absolute differences.

Notice that a "baseline" is shown in the table right after the "/" symbol. As shown in this table, the baseline of "$Rel$" means that the RelTrust approach is chosen as the baseline approach to be compared with other approaches, including CircuitTrust and FlowTrust. Also, the baseline of "$b$" means that the B-FlowTrust algorithm is chosen to be compared with other algorithms, including E-FlowTrust and G-FlowTrust. For example:

$$T_{Cir/Rel}=\sum_{i=1}^{\#RUNS} \frac{ABS(T^i_{Cir}-T^i_{Rel})}{\#RUNS},$$

where $ABS$ is the absolute value function, and $i$ stands for the $i$th round of simulation runs.

TABLE 2
Simulation parameters for Scenario II

| Parameter | Description | Value |
|---|---|---|
| #RUNS | Number of simulation runs | 200 |
| $\mathrm{T_{Cir/Rel}}$ | Avg. of abs. diffs between $\mathrm{T_{Cir}}$ and $\mathrm{T_{Rel}}$ | [0, 1] |
| $\mathrm{T_{b/Rel}}$ | Avg. of abs. diffs between $\mathrm{T_b}$ and $\mathrm{T_{Rel}}$ | [0, 1] |
| $\mathrm{T_{e/Rel}}$ | Avg. of abs. diffs between $\mathrm{T_e}$ and $\mathrm{T_{Rel}}$ | [0, 1] |
| $\mathrm{T_{g/Rel}}$ | Avg. of abs. diffs between $\mathrm{T_g}$ and $\mathrm{T_{Rel}}$ | [0, 1] |
| $\mathrm{T_{e/b}}$ | Avg. of abs. diffs between $\mathrm{T_e}$ and $\mathrm{T_b}$ | [0, 1] |
| $\mathrm{T_{g/b}}$ | Avg. of abs. diffs between $\mathrm{T_g}$ and $\mathrm{T_b}$ | [0, 1] |
| $\overline{\mathrm{C}}\mathrm{T_{e/b}}$ | Avg. of abs. diffs between $\overline{\mathrm{C}}\mathrm{T_e}$ and $\overline{\mathrm{C}}\mathrm{T_b}$ | [0, 1] |
| $\overline{\mathrm{C}}\mathrm{T_{g/b}}$ | Avg. of abs. diffs between $\overline{\mathrm{C}}\mathrm{T_g}$ and $\overline{\mathrm{C}}\mathrm{T_b}$ | [0, 1] |

### 5.3 Simulation Results

We first make two tables to present simulation results for Scenario I. Then, we draw two figures for Scenario II.

Table 3 compares the inferred trust values using RelTrust, CircuitTrust, and FlowTrust. In all our simulation settings, RelTrust infers the largest trust values. In addition, in most of our simulation settings, FlowTrust infers the smallest trust values. That is, CircuitTrust infers the trust values in between RelTrust and FlowTrust in most cases. The reason for RelTrust to always obtain the largest trust values is that the network reliability model assumes that a system is operational if there is at least one operational path between two terminals. This is relatively easy to satisfy since our example trusted graphs have at least two paths in between a trustor and a trustee. Regarding CircuitTrust and FlowTrust, we can observe that CircuitTrust can obtain larger trust values than FlowTrust in most cases, but there are some exceptions.

TABLE 3
Comparison of inferred trust values using RelTrust, CircuitTrust, and FlowTrust

| #N | #E | T-THR | MaxP | RelTrust | CircuitTrust | FlowTrust | | |
|---|---|---|---|---|---|---|---|---|
| | | | | $\mathrm{T_{Rel}}$ | $\mathrm{T_{Cir}}$ | $\mathrm{T_b}$ | $\mathrm{T_e}$ | $\mathrm{T_g}$ |
| 4 | 5 | LOW | 2 | 0.82 | 0.74 | 0.58 | 0.58 | 0.58 |
| 4 | 5 | HIGH | 2 | 0.92 | 0.81 | 0.85 | 0.85 | 0.85 |
| 8 | 12 | LOW | 2 | 0.80 | 0.73 | 0.63 | 0.36 | 0.61 |
| 8 | 12 | HIGH | 3 | 1.00 | 0.95 | 0.87 | 0.76 | 0.87 |
| 12 | 18 | LOW | 2 | 0.80 | 0.73 | 0.74 | 0.42 | 0.59 |
| 12 | 18 | HIGH | 2 | 0.98 | 0.83 | 0.82 | 0.41 | 0.82 |
| 16 | 24 | LOW | 2 | 0.89 | 0.72 | 0.74 | 0.47 | 0.59 |
| 16 | 24 | HIGH | 2 | 0.98 | 0.85 | 0.89 | 0.46 | 0.80 |
| 20 | 30 | LOW | 2 | 0.73 | 0.55 | 0.64 | 0.36 | 0.58 |
| 20 | 30 | HIGH | 3 | 1.00 | 0.89 | 0.85 | 0.64 | 0.84 |

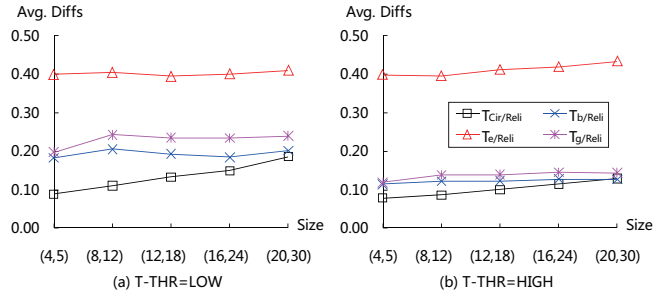


Fig. 6. Statistical results on the inferred trust values using RelTrust, CircuitTrust, and FlowTrust approaches.

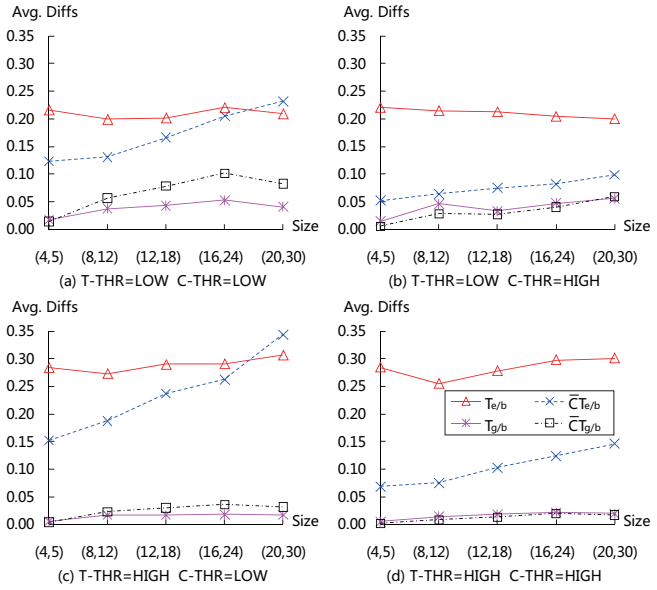


Fig. 7. Statistical results on the normalized trust metrics using our three FlowTrust algorithms.

Table 4 compares the normalized trust metrics using B-FlowTrust, E-FlowTrust, and G-FlowTrust. We can observe that B-FlowTrust always infers the largest normalized trust values (and also the largest normalized confidence levels) among these three algorithms. This is in accordance with our Theorem 4. We can also observe that G-FlowTrust always infers all the four normalized trust metrics which are close to B-FlowTrust. This shows that the graph simplification-based algorithm does work in our FlowTrust approach, although such an algorithm may lose some information due to the fact that some trust relationships will not be taken into account in the trust inference. However, we point out that this works well only when we can find a set of disjoint paths in a trusted graph that can maximize the inferred trust value (and also the inferred confidence level). In our simulations, we used the exhaustive enumeration to find such a set, however, it is very time consuming.

Fig. 6 shows the statistical results of the inferred trust values using RelTrust, CircuitTrust, and FlowTrust approaches. All the statistical results are obtained by averaging on all the simulation runs of $\#RUNS = 200$. Our comparison used RelTrust as the baseline approach since it always infers the largest trust values among the three approaches. The X-axis shows the size of a trusted graph, indicated by the number of nodes $(\#N)$ and the number of edges $(\#E)$. The Y-axis shows the average value of the absolute differences between different approaches, *Avg. Diffs* for short. We can observe that there is an apparent gap between any pair of these approaches. Basically, this means that different approaches are comparable to each other, although their inferred trust values are different from each other. To put it another way, if we remove the gap between a pair of these approaches, then we can show their equivalence between each other.

Fig. 7 shows the statistical results of the normalized trust metrics using our three FlowTrust algorithms. All the statistical results are obtained by averaging on all the simulation runs of $\#RUNS = 200$. Our comparison used B-FlowTrust as the baseline algorithm since it always infers the largest normalized trust values (and also the largest normalized confidence levels) among the three algorithms. The X-axis shows the size of a trusted graph, and the Y-axis shows *Avg. Diffs*, as introduced above. Since each algorithm infers four normalized trust metrics, we can obtain eight such metrics for E-FlowTrust and G-FlowTrust algorithms. We only report here four metrics, i.e., $T_e$, $\overline{C}T_e$, $T_g$, and $\overline{C}T_g$, since the other four metrics have the same trend as observed from these four metrics.

From all the simulation settings in Fig. 7, we can observe that G-FlowTrust infers all the normalized trust metrics that are close to B-FlowTrust. So, G-FlowTrust can be considered a good approximation of B-FlowTrust. The problem is that we need first to find the maximum number of edge-disjoint paths in a trusted graph that also maximizes the normalized trust values (also the normalized confidence levels). This is not a trivial problem. So, we used the exhaustive enumeration in our simulations for illustrative purposes, which is quite time consuming. In addition, we can also observe an apparent gap between E-FlowTrust and B-FlowTrust in this figure. This gap shows that E-FlowTrust may not be a good approximation of B-FlowTrust in the general case. However, we point out that it would be a good approximation if all the node degrees in a trusted graph are the same (or similar), and they are also the same (or similar) with the maximum number of edge-disjoint paths (See Theorem 1). All in all, the simulation results can reflect the availability of the algorithms to a great extent, although the real-world experiments are very important for the validation of new algorithms. Also, by comparison with CircuitTrust and RelTrust, we can find that FlowTrust is a relatively conservative approach and suitable for pessimistic users that are prone to put low trust on others.

TABLE 4
Comparison of normalized trust metrics using B-FlowTrust, E-FlowTrust, and G-FlowTrust

| #N | #E | T-THR | C-THR | MaxP | B-Flow Trust | | | | E-Flow Trust | | | | G-Flow Trust | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | $T_b$ | $C_b$ | $\overline{CT}_b$ | $\overline{TC}_b$ | $T_e$ | $C_e$ | $\overline{CT}_e$ | $\overline{TC}_e$ | $T_g$ | $C_g$ | $\overline{CT}_g$ | $\overline{TC}_g$ |
| 4 | 5 | LOW | LOW | 2 | 0.64 | 0.81 | 0.15 | 0.48 | 0.43 | 0.54 | 0.10 | 0.32 | 0.64 | 0.80 | 0.15 | 0.48 |
| 4 | 5 | LOW | HIGH | 2 | 0.60 | 0.86 | 0.15 | 0.58 | 0.40 | 0.57 | 0.10 | 0.38 | 0.60 | 0.80 | 0.16 | 0.51 |
| 4 | 5 | HIGH | LOW | 2 | 0.80 | 0.69 | 0.40 | 0.26 | 0.53 | 0.46 | 0.26 | 0.17 | 0.80 | 0.69 | 0.40 | 0.26 |
| 4 | 5 | HIGH | HIGH | 2 | 0.86 | 0.88 | 0.18 | 0.20 | 0.57 | 0.59 | 0.12 | 0.13 | 0.86 | 0.85 | 0.18 | 0.19 |
| 8 | 12 | LOW | LOW | 2 | 0.63 | 0.52 | 0.40 | 0.33 | 0.36 | 0.42 | 0.23 | 0.29 | 0.61 | 0.52 | 0.37 | 0.33 |
| 8 | 12 | LOW | HIGH | 2 | 0.64 | 0.84 | 0.18 | 0.63 | 0.36 | 0.44 | 0.11 | 0.32 | 0.57 | 0.82 | 0.13 | 0.60 |
| 8 | 12 | HIGH | LOW | 3 | 0.87 | 0.61 | 0.51 | 0.13 | 0.76 | 0.51 | 0.47 | 0.10 | 0.87 | 0.61 | 0.51 | 0.13 |
| 8 | 12 | HIGH | HIGH | 2 | 0.83 | 0.84 | 0.32 | 0.33 | 0.41 | 0.42 | 0.16 | 0.17 | 0.83 | 0.84 | 0.32 | 0.33 |
| 12 | 18 | LOW | LOW | 2 | 0.74 | 0.79 | 0.61 | 0.66 | 0.42 | 0.49 | 0.32 | 0.43 | 0.59 | 0.71 | 0.47 | 0.59 |
| 12 | 18 | LOW | HIGH | 2 | 0.61 | 0.88 | 0.27 | 1.03 | 0.36 | 0.47 | 0.18 | 0.59 | 0.54 | 0.84 | 0.23 | 0.94 |
| 12 | 18 | HIGH | LOW | 2 | 0.82 | 0.80 | 0.81 | 0.37 | 0.41 | 0.40 | 0.40 | 0.19 | 0.82 | 0.68 | 0.81 | 0.23 |
| 12 | 18 | HIGH | HIGH | 2 | 0.87 | 0.82 | 0.33 | 0.32 | 0.45 | 0.41 | 0.17 | 0.16 | 0.86 | 0.82 | 0.33 | 0.32 |
| 16 | 24 | LOW | LOW | 2 | 0.74 | 0.62 | 0.75 | 0.56 | 0.47 | 0.45 | 0.52 | 0.52 | 0.59 | 0.61 | 0.51 | 0.48 |
| 16 | 24 | LOW | HIGH | 2 | 0.58 | 0.83 | 0.32 | 0.96 | 0.32 | 0.45 | 0.15 | 0.48 | 0.57 | 0.83 | 0.31 | 0.96 |
| 16 | 24 | HIGH | LOW | 2 | 0.89 | 0.50 | 1.42 | 0.26 | 0.46 | 0.25 | 0.75 | 0.13 | 0.80 | 0.50 | 1.24 | 0.26 |
| 16 | 24 | HIGH | HIGH | 3 | 0.82 | 0.85 | 0.33 | 0.39 | 0.75 | 0.78 | 0.40 | 0.45 | 0.82 | 0.83 | 0.32 | 0.36 |
| 20 | 30 | LOW | LOW | 2 | 0.64 | 0.54 | 0.92 | 0.72 | 0.36 | 0.35 | 0.52 | 0.48 | 0.58 | 0.54 | 0.74 | 0.72 |
| 20 | 30 | LOW | HIGH | 2 | 0.61 | 0.80 | 0.47 | 1.46 | 0.35 | 0.44 | 0.26 | 0.79 | 0.54 | 0.80 | 0.41 | 1.46 |
| 20 | 30 | HIGH | LOW | 3 | 0.85 | 0.67 | 0.59 | 0.27 | 0.64 | 0.55 | 0.44 | 0.22 | 0.84 | 0.63 | 0.57 | 0.23 |
| 20 | 30 | HIGH | HIGH | 3 | 0.85 | 0.83 | 0.40 | 0.46 | 0.71 | 0.73 | 0.40 | 0.50 | 0.83 | 0.80 | 0.38 | 0.42 |

## 6 Conclusion

Trust has been studied for many years, particularly in P2P networks [23] and mobile ad-hoc networks [1]. Trust, in social networks, is relatively new. But, trust becomes more and more important due to the rapid development of social networking applications in recent years. Our proposed FlowTrust is quite easy to understand, and it can also support multi-dimensional trust in social networks. With the proposed approach, we can deduce trust in web-based social networks, P2P networks, and many others. The inferred trust results will provide important indications for related applications, such as recruiting trustworthy employees from social networks, and purchasing products from trustworthy online stores. Of course, this approach is only one trust evaluation method, and can be integrated into any trust management systems. Basically, FlowTrust can efficiently deal with small trusted graphs since the network flow algorithms may not be efficient for large networks. However, most social networks are large networks, so, there is a gap between large social networks and small trusted graphs. We argue that this gap can be overcome by developing algorithms to deduce small graphs from large networks, based on existing theoretical results, such as the *Six Degrees of Separation* phenomenon [22]. According to this phenomenon, in most cases, the maximum path length between a trustor $u$ and a trustee $v$ can be confined to six. Currently, we are working on how to deduce a small trusted graph that is adequate enough for trust inference from a large social network. Here, "adequate enough" means to achieve similar "fairness" in node/edge-disjoint multiple paths when selecting nodes and edges to form a trusted graph.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant Nos. 90718034, 61073037 & 60773013, the Hunan Provincial Natural Science Foundation of China for Distinguished Young Scholars under Grant No. 07JJ1010, and the Changsha Science and Technology Program under Grant Nos. K1003064-11 & K1003062-11.

## References

[1] T. Anantvalee and J. Wu, *Reputation-Based System for Encouraging the Cooperation of Nodes in Mobile Ad Hoc Networks*, Proc. of IEEE ICC 2007, pp. 3383C3388, June 2007.
[2] C. J. Colbourn, *The Combinatorics of Network Reliability*, J. Hopcroft, Ed., ISBN 978-0195049206, Oxford University Press, 1987.
[3] Facebook, *http://www.facebook.com/*.
[4] P. Flocchini and F. L. Luccio, *Routing in Series Parallel Networks. Theory of Computing Systems*, 36(2): 137-157, 2003.
[5] D. Gefen, *Reflections on the Dimensions of Trust and Trustworthiness among Online Consumers*, ACM SIGMIS Database, 33(3): 38-53, August 2002.
[6] J. Golbeck and J. Hendler, *Inferring Binary Trust Relationships in Web-Based Social Networks*, ACM Transactions on Internet Technology, 6(4): 497-529, November 2006.
[7] N. Griffiths, *Task Delegation Using Experience-Based Multi-Dimensional Trust*, Proc. of AAMAS 2005, pp. 489-496, July 2005.
[8] L. Guanfeng, W. Yan, and M. Orgun, *Trust Inference in Complex Trust-Oriented Social Networks*, Proc. of IEEE CSE 2009, pp.29-31, August 2009.
[9] W. H. Hayt, J. E. Kemmerly, and S. M. Durbin, *Engineering Circuit Analysis*, Seventh Edition, ISBN 978-0471407409, New York: McGraw-Hill Higher Education, February 2007.
[10] A. Hnativ and S. A. Ludwig, *Evaluation of Trust in an eCommerce Multi-agent System Using Fuzzy Reasoning*, Proc. of the IEEE 18th international conference on Fuzzy Systems, pp.20-24, August 2009.
[11] A. Jøsang, R. Hayward, and S. Pope, *Trust Network Analysis with Subjective Logic*, Proc. of ACSC 2006, pp. 85-94, January 2006.
[12] LinkedIn, *http://www.linkedin.com/*.
[13] G. Mahoney, W. Myrvold, and G. C. Shoja, *Generic Reliability Trust Model*, Proc. of PST 2005, pp. 113-120, October 2005.

[14] L. Mui, M. Mohtashemi, and A. Halberstadt, *A Computational Model of Trust and Reputation*, Proc. of HICSS 2002, pp. 2431-2439, January 2002.
[15] MySpace, *http://www.myspace.com/*.
[16] M. K. Reiter and S. G. Stubblebine, *Resilient Authentication Using Path Independence*, IEEE Transactions on Computers, 47(12): 1351-1362, December 1998.
[17] D. P. Siewiorec and R. S. Swarz, *Reliable Computer Systems: Design and Evaluation*, Third Edition, ISBN 978-1568810928, AK Peters, Ltd., October 1998.
[18] 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 Journal on Selected Areas in Communications, 24(2): 305-317, February 2006.
[19] M. Taherian, M. Amini, and R. Jalili, *Trust Inference in Web-Based Social Networks using Resistive Networks*, Proc. of ICIW 2008, pp. 233-238, June 2008.
[20] G. Theodorakopoulos and J. S. Baras, *On Trust Models and Trust Evaluation Metrics for Ad Hoc Networks*, IEEE Journal on Selected Areas in Communications, 24(2): 318-328, February 2006.
[21] G. Wang and J. Wu, *Multi-Dimensional Evidence-Based Trust Management with Multi-Trusted Paths*, Future Generation Computer Systems (Elsevier), Published Online on May 7, 2010, DOI: http://dx.doi.org/10.1016/j.future.2010.04.015.
[22] D. J. Watts, *Six Degrees: The Science of a Connected Age*, ISBN 978-0393325423, W. W. Norton & Company, February 2004.
[23] L. Xiong and L. Liu, *PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities*, IEEE Transactions on Knowledge and Data Engineering, 16(7): 843-857, July 2004.
[24] Y. Zuo, W.-C. Hu, and T. O'Keefe, *Trust Computing for Social Networking*, Proc. of ITNG 2009, pp. 1534-1539, April 2009.

**Guojun Wang** received a B.Sc. in Geophysics, a M.Sc. in Computer Science, and a Ph.D. in Computer Science from Central South University (CSU), China. Since 2005, he is a Professor at CSU. Since 2006, he is a doctoral supervisor at CSU. He is the Director of Trusted Computing Institute of CSU. He is a Vice Head of Department of Computer Science and Technology of CSU. He has been an Adjunct Professor at Temple University, USA, a Visiting Scholar at Florida Atlantic University, USA, a Visiting Researcher at the University of Aizu, Japan, and a Research Fellow at the Hong Kong Polytechnic University, Hong Kong. His research interests include trusted computing, mobile computing, pervasive computing, and software engineering. He is a member of IEEE, and a senior member of CCF (China Computer Federation).

**Jie Wu** is Chair and Professor at the Department of Computer and Information Sciences at Temple University. He is an IEEE Fellow. He is on the editorial board of IEEE Transactions on Mobile Computing. He was a Distinguished Professor in the Department of Computer Science and Engineering, Florida Atlantic University. He served as a Program Director at US NSF from 2006 to 2008. He has been on the editorial board of IEEE Transactions on Parallel and Distributed Systems. He has served as a distinguished visitor of the IEEE Computer Society and is the chairman of the IEEE Technical Committee on Distributed Processing (TCDP). His research interests include wireless networks and mobile computing, parallel and distributed systems, and fault-tolerant systems.