Bounding Trust in Reputation Systems with Incomplete Information

Xi Gong
North Carolina State University
xgong2@ncsu.edu

Ting Yu
North Carolina State University
tyu@csc.ncsu.edu

Adam J. Lee
University of Pittsburgh
adamlee@cs.pitt.edu

ABSTRACT

Reputation mechanisms represent a major class of techniques for managing trust in decentralized systems. Quite a few reputation-based trust functions have been proposed in the literature for use in many different application domains. However, in many situations, one cannot always obtain all of the information required by the trust evaluation process. For example, access control restrictions or high collection costs might limit one's ability to gather every possible feedback that could be aggregated. Thus, one key question is how to analytically quantify the *quality* of reputation scores computed using *incomplete information*.

In this paper, we start a first effort towards answering the above question by studying the following problem: given the existence of certain missing information, what are the worst and best trust scores (i.e., the bounds of trust) a target entity can be assigned by a given reputation function? We formulate this problem based on a general model of reputation systems, and then examine the ability to bound a collection representative trust functions in the literature. We show that most existing trust functions are monotonic in terms of direct missing information about the target of a trust evaluation, which greatly simplifies this process. The problem of trust bounding with the presence of indirect missing information is much more complicated. We show that many well-known trust functions are not monotonic regarding indirect missing information, which means that a case-by-case analysis needs to be conducted for each trust function in order to bound an entity's trust.

Categories and Subject Descriptors: C.2.4 [Computer-Communication Networks]: Distributed Systems—*Distributed applications*; K.4.4 [Computer and Society]: Electronic Commerce—*Security*

General Terms: Security

Keywords: Reputation, trust, missing information

CODASPY'12, February 7–9, 2012, San Antonio, Texas, USA.

1. INTRODUCTION

Large-scale decentralized systems—e.g., P2P networks and online auction communities—allow entities from different security domains to interact and conduct business with each other in an ad-hoc manner. Such systems offer significant advantages in terms of service variety, availability, and robustness. As a priori trust relationships do not typically exist between entities, establishing trust at runtime is a key problem in the design of trustworthy decentralized systems. Reputation mechanisms are a prominent technique for trust management in these types of systems. In a reputation system, once a transaction is finished, the participants issue feedback that evaluates the service or behavior of one another during the transaction. Before a new transaction starts, one may first assess an entity's trustworthiness based on the feedback provided by other entities. This process can be viewed as the application of a so-called *trust function* that takes as input the feedbacks from an entity's past transactions (and possibly those of other related parties), and outputs a trust value to indicate its trustworthiness.

Quite a few trust functions have been proposed in the literature. Some are based on generic models of decentralized systems, while some others target specific applications domains. The designs of these trust functions may thus differ greatly with respect to methodologies for trust inferences as well as protocols for information collection. However, existing work largely overlooks the problem of missing information and its impact on trust evaluation. In large-scale decentralized systems, there are many reasons that one may not always be able to access all the information required by a trust function. For example, some entities may impose access control restrictions regarding how feedbacks they issue can be accessed by others in order to mitigate potential privacy concerns. In other situations, especially in P2P networks, feedbacks are stored in a distributed manner among multiple entities. Some entities may not be online at the time of trust evaluation, and thus the feedbacks stored by these entities may be unavailable. Sometimes, even if all necessary feedbacks are retrievable, it may be too costly to collect them all. As a result, it is not uncommon to evaluate an entity's trust based upon incomplete information.

A natural question is thus how to quantify the quality of an entity's trustworthiness score when it is computed using incomplete information. There are several ways to formulate the above question. For example, if we can make reasonable assumptions about the distribution of the missing feedback information, we could perhaps compute the 95% confidence interval for a participant's trustworthiness. In this paper,

we study a deterministic version of this problem that makes no assumptions regarding the distribution of missing information: given the range of possible values taken by missing information, what are the possible trust scores for an entity? In particular, we are interested in establishing the worst- and best-possible trustworthiness ratings for a target entity.

Answering this problem can help us make better decisions in decentralized systems. For instance, suppose our policy is to conduct a transaction with an entity if its trustworthiness is over 0.8. If the bounds of its trustworthiness can be established as $[0.8, 0.9]$, then it is safe to do business with that entity, even if some information is missing. For sampling based approaches to dealing with missing information, this bound can guide us to decide whether it is appropriate to stop sampling. A wide bound (e.g., $[0.2, 1]$) suggests the current degree of sampling is insufficient, resulting in the large uncertainty about the target's trustworthiness. On the other hand, a narrow bound (e.g., $[0.85, 0.9]$) indicates diminishing returns for further sampling. This paper presents our first effort towards exploring the above trust bounding problem. Our contributions can be summarized as follows:

- We present a general model of reputation assessment, categorize the possible types of missing information, and formally define the trust bounding problem.
- We develop the notion of *monotonicity* as it relates the evaluation of trust functions using incomplete information. This provides a generic method for bounding an entity's trustworthiness given any monotonic trust function. We further argue that monotonicity regarding direct missing information should be a property of any reasonable trust function.
- We study the monotonicity of a large set of representative trust functions from the literature. We find that, technically, monotonicity proofs for trust functions based on weighted recommendations or direct topology information are relatively easy. On the other hand, monotonicity proofs for trust functions based on matrix iterations are particularly challenging.
- We show that monotonicity in general does not hold for indirect missing information by providing concrete examples of existing trust functions in the literature. This suggests that the trust bounding problem has to be studied in a case-by-case manner for specific trust functions.

The rest of this paper is organized as follows. In Section 2, we give a general model of reputation systems, categorize the possible types of missing information, and formally define the trust bounding problem. In Section 3, we propose a classification of trust functions based on their design principles. Section 4 then introduces the monotonicity property of trust functions and discusses its relation to trust bounding. In Sections 5, we analyze the monotonicity of several representative trust functions with respect to a single missing edge with known topology in the trust graph. We report on related work in Section 6, and conclude in Section 7.

2. PROBLEM DEFINITION

In this section, we first provide a general model for reasoning about reputation systems. We then categorize the types of missing information that may exist within such a system. Finally, we formalize the trust bounding problem.

2.1 A General Model For Reputation System

In decentralized systems, entities interact with each other through transactions, which may include monetary interactions, retrieving information from a website, file downloads, and the like. Without loss of generality, we assume that a transaction is uni-directional, i.e., there is a clear distinction between a service provider and a service consumer. Note that a service provider in one transaction may be a consumer in another transaction. A *feedback* is a report issued by a consumer about a provider after a transaction. It can be denoted as a four-tuple (c, s, r, t), indicating that consumer c issues a feedback about provider s with a rating r at time t. In practice, a rating r may be multi-dimensional, covering multiple aspects (e.g., product quality, customer service, shipping speed, etc.) of a transaction. The value for a rating may also be taken from any total ordered domain. Some examples include 0 to 5 stars, a binary positive/negative vote, or a choice from a scale ranging from poor to excellent. For simplicity, in this paper we focus on binary positive/negative votes; as we will see later, such a setting is compatible with the design of most existing trust functions.

Though multiple transactions may happen between a pair (c, s) of consumer and provider, most trust functions in the literature consider only an aggregate of these transactions (e.g., the ratio of positive or negative transactions among all the transactions). We call such an aggregation the *opinion* of c over s, or c's *local trust* of s. The information in a reputation system can be modeled as a *trust graph*.

DEFINITION 1 (TRUST GRAPH). *Given a set of principals P and a totally ordered domain of weights W, a* trust graph *is a weighted directed graph $G = \langle P, E \rangle$. The edge (c, s, w) in the set $E \subseteq P \times P \times W$ encodes the local opinion w that the consumer c has for the service provider s.*

Figure 1 shows an example trust graph.

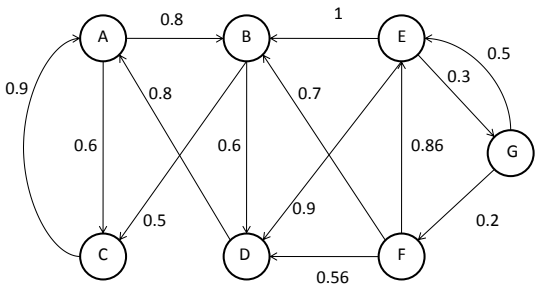


Figure 1: An Example Trust Graph

Although the specific design of trust functions in the literature may differ significantly, most can be abstracted into a common form.

DEFINITION 2 (TRUST FUNCTION). *Given a trust graph $G = \langle P, E \rangle$ and a totally ordered domain W, a* trust function *$F : G \times P \times P \to W$ is a function that takes as input trust graph and a pair of entities u and v, and computes the extent of u's trust in v.*

Intuitively, given the feedback information stored in a trust graph, F computes the trustworthiness of v from u's point of view. For some trust functions (e.g., [5]) $F(G, u, v) =$

$F(G, u', v)$ for any $u, u' \in P$. In general, however, it is often the case that $F(G, u, v) \neq F(G, u', v)$ if $u \neq u'$.

2.2 Types of Missing Information

There can be several types of missing information within the trust graph observed by a principal in the system. For instance, there may be one or more missing feedbacks between one or more pairs of entities. Alternately, there may be entire pieces of the topology that are unknown. To help make a more systematic analysis later, we now categorize the possible missing information scenarios.

- **Single missing edge weight, known topology.** Having known topology of the trust graph means that we can explicitly enumerate all of the possible paths from any pair of nodes in the trust graph. In this scenario, missing information is restricted to some subset of the feedbacks between one pair of principals, which results in a missing edge weight. This missing information may be due, for example, to restrictions imposed by individual principals. For instance, we may know that there are a total of 10 transactions between Alice and Bob, but Alice only makes the feedbacks of 6 of these interactions accessible. As a result, we cannot know Alice's *exact* opinion of Bob.
- **Set of missing edges, known topology.** This situation is a generalization of the above single-edge case. This may occur, for instance, as a result of a principal's access controls. For instance, if Bob requests feedback reports from Alice, she may provide him with a subset of her reports for multiple principals. Thus Bob might know everyone with whom Alice has interacted, but have incomplete weight values for each.
- **Unknown topology.** In this case, not only can edge *weights* be missing, but perhaps entire *edges* may be missing from some principal's view of the trust graph. For example, a subgraph of the complete trust graph G may not be accessible at all due to restrictions imposed by security policies of a domain. Note that missing edges also imply missing weights on those edges.

2.3 The Trust Bounding Problem

Given the above possible types of missing information, we are interested in bounding the possible trust values that can be assigned to some target principal v in the system by some source principal u. We formalize the *trust bounding problem* as follows:

DEFINITION 3 (EXTENDS, $\sqsupseteq$). *Let $\perp$ denote an unknown edge weight. Given two trust graphs $G = \langle P, E\rangle$ and $G' = \langle P', E'\rangle$, we say that G'* extends *G if and only if (i) $P \subseteq P'$ and (ii) $(u, v, w) \in E' \leftrightarrow [(u, v, w) \in E \vee (u, v, \perp) \in E \vee \nexists w' \in W : (u, v, w') \in E]$. We denote by $G' \sqsupseteq G$ the fact that G' extends G.*

Informally, a trust graph G' extends a trust graph G if and only if (i) it contains a superset of G's vertices, and (ii) all edges in G' already exist in G, provide a weight for an edge with an unknown weight in G, or connect two vertices that are unconnected in G.

DEFINITION 4 (TRUST BOUNDING PROBLEM). *Given a partial trust graph $G = \langle P, E\rangle$, a trust function F, a source principal u, and a destination principal v, the* trust bounding problem *is to compute the pair $\langle min_{G' \sqsupseteq G} F(G', u, v), max_{G' \sqsupseteq G} F(G', u, v)\rangle$.*

In this paper, as a first effort, we restrict our discussion to the case of a single missing edge weight in the trust graph. That is, given the range of possible values taken by this single missing edge, what are the possible trust scores for an entity? In particular, we are curious about establishing the worst- and best-possible trustworthiness of the entity.

DEFINITION 5 (SINGLE EDGE TRUST BOUNDING PROBLEM). *Given a trust graph $G = \langle P, E\rangle$, a trust function F, and a single edge $(u, v, w) \in E$ for which the weight w is unknown, the* single edge trust bounding problem *is to compute the pair $\langle min_w F(G, u, v), max_w F(G, u, v)\rangle$.*

3. TYPES OF TRUST FUNCTIONS

We have surveyed a large set of representative trust functions in the literature and have observed that although the specific functions may differ from each other greatly, their underlying design principles can be categorized into three major classes. We now discuss each class and several representative trust functions, which will be helpful during our later analysis of the trust bounding problem.

3.1 Recommendation Based Functions

The idea of this class of trust functions can be explained with the help of the following formula.

$$t_{uv} = \sum_{j \in S} w_{uj} \cdot t_{jv}. \tag{1}$$

When an entity u wants to evaluate the trust of another entity v but has not had any direct interactions with v, she can ask for opinions from a set $S \subseteq P$ of witnesses j who have directly interacted with v. These opinions are denoted as t_{jv}, and are combined using weights w_{uj} which represent u's belief in j's opinion. The following are two representative examples from this class of functions.

Managing the Dynamic Nature of Trust (MDNT) [11]. The trust function in this work can be summarized as the following formula.

$$Rep(u, v) = \sum_{i=1}^{N} WTV(i) \diamondsuit Rep(i, v) + \beta \times \sum_{j=1}^{M} Trustworthiness(j, v)/M. \tag{2}$$

The meaning of the formula is as followed. Suppose entity u want to evaluate the reputation of entity v. Then she asks for opinions from her witness neighborhood which consists of i's. $WTV(i)$ depicts the correctness of entity i in historical recommendations. $\diamondsuit$ allows some flexibility in the choice of a concatenation function. In addition, u collects opinions from M entities whose WTV's is unknown to her. So she assigns them the same weight in the recommendation, reflected by taking an arithmetic mean of the opinions from these M entities. Finally, a positive changeable parameter β is used to adjust the impact of these entities. In this function, WTV plays an important role in generating the weights of the witnesses. After each transaction, the source entity s rates the quality of the service from t and compares it with the

recommendations received from the witness entities. The difference is stored for the witness's record. After a long time period, many records are collected to evaluate whether a witness entity is a good recommender, and the evaluation result is represented by WTV.

Credence [12]. Credence is a distributed reputation mechanism for P2P file sharing systems. In Credence, users manually provide positive/negative votes indicating the authenticity of a downloaded file. Historical records of an entity's recent votes are stored. In this context, we can consider a file as the v entity in a feedback (u, v, r, t). When a file's authenticity or quality needs to be evaluated, a vote is raised among the neighbors of u. The final trust for the file is computed by taking a weighted average of the neighbor entities' opinions.

The weights of the neighbors depend on the statistical correlation between the vote history of u and that of each of its neighbors, which is computed using the following formula:

$$\theta_{uv} = (P - UV)/\sqrt{U(1-U)V(1-V)}. \quad (3)$$

where U (respectively V) is the fraction of votes from u (respectively v) with positive intention, and P is the fraction of such pairs that agree with both votes having positive intention. θ is called coefficient of correlation, and takes value in $[-1, 1]$. It's not hard to prove that a value of θ being close to 1 implies a strong agreement between the pair and a value close to -1 indicates a strong disagreement between the pair. Moreover, when θ_{uv} is close to 0, the pair can be considered as being irrelative or independent. Then, θ_{uv} can be taken in formula 1 to compute weight.

$$t_{uv} = \sum_{j \in S} \frac{\theta_{jv}}{\sum_j \theta_{jv}} \cdot Vote_{jv}. \quad (4)$$

where S is the set of voters.

As we can see from the above two examples, the key to this type of trust functions lies in the definition of weights. In [11] weights are computed based upon the correctness of historical recommendations made by a witness, while in Credence the weights depend on the statistical correlation between the voting history of the source entity and that of the witness. A key observation is that a specific opinion is used either in computing a witness' weight(WTV and θ) or in the opinion of a witness(Rep, $Trustworthiness$ and $Vote$), but not both. As we will see, this property allows us to derive a general method to reason the monotonicity of this type of trust functions and simplify the trust bounding problem.

3.2 Topology Based Functions

This class of trust functions makes explicit use of the topology of a trust graph when calculating trust score. When an entity u wants to evaluate another entity v with whom it has not interacted with directly, it finds paths from u to v in G. A concatenation algorithm is then used to derive the strength of these paths. The final trust is achieved from a weighted aggregation over all such paths. This class of functions bears some similarity to the class of recommendation based trust functions. Nevertheless, the pivotal difference is that the weight on a single edge may contribute to both the strength of a path and the weight of that path in an aggregation. Representative examples of this class of trust functions include the trust function proposed in the NICE system [6] and the Beta Reputation System [3].

NICE [6]. After each transaction in the NICE system, the client u signs a cookie stating the quality of the transaction with the server v, which later can be used to prove the trustworthiness of v to others. Two functions are then proposed to calculate the trust that u should have for v. One function computes the weight of the strongest path, while the other function computes the weighted sum of the strongest disjoint paths from u to v. In the process of inferring trust along a path, the minimum weight of any edge along the path is taken as the strength of the path. If the "strongest path" function is used, it is obvious to find the path with the highest score. If instead the "weighted sum" function is used, the weight on the first-hop edge starting from u is defined as the weight of the path.

Beta Reputation Systems[3]. The Beta Reputation System is another standard topology based trust function. It assumes that the feedback for every transaction is a binary outcome $\{1, 0\}$, where 1 represents a positive feedback and 0 represents a negative feedback. It first aggregates feedbacks locally to get two inputs for an entity v. r_v^u represents the amount of positive feedbacks for v provided by u, while s_v^u represents the amount of negative feedbacks for u. The quantities r_v^u and s_v^u are then used as shape parameters for a beta distribution. By plugging them in $\varphi(p|r_v^u, s_v^u)$ we get

$$\varphi(p|r_v^u, s_v^u) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{(\alpha-1)}(1-p)^{(\beta-1)}, \quad (5)$$
$$0 \le p \le 1, \alpha > 0, \beta > 0.$$

Then it derives the expected value from the distribution and converts it into a more straightforward expression.

$$(E(\varphi(p|r_v^u, s_v^u)) - 0.5) * 2 = \frac{r_v^u - s_v^u}{r_v^u + s_v^u + 2} \in [-1, 1]. \quad (6)$$

where $E(\varphi(p|r_v^u, s_v^u)) = \frac{r_v^u + 1}{r_v^u + s_v^u + 2}$ is the expected value.

Moreover, the author proposes specific formulas to aggregate and concatenate opinions from multiple entities. When an entity wants to evaluate the trust of another entity,it can compute the trust score of the target following the aggregation and concatenation formulas provided that it can find some paths to the target entity through the trust graph.

3.3 Matrix Based Functions

This final class of trust functions take the adjacency matrix of a trust graph as input, and iteratively apply various matrix operators until convergence to derive each entity's trust. The Path Algebra [9], PeerTrust [13], and EigenTrust [5] schemes are all representative examples of this class of trust functions.

Path Algebra [9]. Path Algebra assumes that feedbacks have been aggregated into opinions in the range $[0, 1]$. The basic intuition is then similar to that of NICE. Specifically, Path Algebra defines two basic functions: concatenation and aggregation. A concatenation function computes the strength of path from the source u to the target v (intuitively it concatenates the opinion on each edge of the path to form the strength of the whole path). And an aggregation function aggregates the strengths of multiple paths to form the final trust. The choice of these functions can be left open to specific applications. Typical concatenation functions include multiplication and minimization, while typical aggregation functions can be maximization, minimization, and average.

Unlike NICE—which explicitly enumerates paths from u to v—Path Algebra requires repetitive combination of the concatenation and aggregation functions until the result converges. This is achieved through a process similar to matrix multiplication. Specifically, given a row $(a_1, \ldots, a_n)$ and a column $(b_1, \ldots, b_n)$ of a matrix, Path Algebra first computes $c_i = C(a_i, b_i)$ where C is the concatenation function, and then aggregates all c_i using the aggregation function. When the combination of minimization (concatenation) and maximization (aggregation) is chosen, the computation is equivalent to finding a strongest path between each pair of entities. As an instance, Zhao et al implemented VectorTrust [15] which uses maximization to aggregate and multiplication to concatenate.

PeerTrust [13]. PeerTrust derives a matrix $\mathbf{A}$ that is based on complaint records reported by entities in the system.

$$\mathbf{A} = (a_{v,u}) = \begin{cases} \frac{C(v,u)}{I(v)} & \text{if } I(v) \neq 0 \\ 0 & \text{if } I(v) = 0 \end{cases} \quad (7)$$

In the above equation, $C(v, u)$ denotes the aggregate number of complaints (negative feedbacks) that entity v receives from entity u, and $I(v)$ represents the amount of transactions that v has. PeerTrust then executes the following operations on the above matrix, where $t_1 = (1, 1, \ldots, 1)^T$:

$$\text{Repeat: } t_{n+1} = \mathbf{1} - \mathbf{A}t_n;$$
$$\text{Until we get a converged } t_n.$$

In the converged vector, each element corresponds to the trust of an entity, independent of the source of the trust evaluation. As such, we call the resulting trust scores *global trust scores*.

EigenTrust [5]. In EigenTrust, a user u rates her transaction with service provider v as either positive ($tr(v, u) = 1$) or negative ($tr(v, u) = -1$). Then EigenTrust defines

$$s_{vu} = \text{ sum of } tr(v,u)\text{'s} = sat(v,u) - unsat(v,u);$$

$$c_{vu} = \frac{max(s_{vu}, 0)}{\sum_u max(s_{vu}, 0)};$$

where $sat(v, u)$ and $unsat(v, u)$ are the number of satisfactory and unsatisfactory transactions that v received from u respectively.

Moreover, EigenTrust computes a matrix $\mathbf{C} = [c_{vu}]$ and performs the following operations on this matrix:

$$\text{Repeat: } t_{n+1} = \mathbf{C}^T t_n;$$
$$\text{Until we get a converged } t_n.$$

where n denotes the number of participants in the community and $t_1 = (\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n})^T$.

EigenTrust computes a global trust score for each entity. However, unlike PeerTrust, it ensures a normalization of trust. That is at each step, EigenTrust ensures that the summation of all the elements in t equals 1.

4. MONOTONICITY DEFINITION

Clearly, trust bounding is a nontrivial problem. Enumerating all possible weight assignments for missing edge values would be too expensive, especially in the event of a continuous weight domain. To simplify this process, we now explore a property of certain trust functions that we call *monotonicity*. We focus, in particular, on monotonicity with respect to a single edge variable.

Definition 6 (Monotonicity, Single Edge Case). *Consider a trust function F. Let G be a trust graph with a single undefined edge $(x, y, \perp)$, and let $G(w)$ denote the extension of G with the edge (x, y, w). Given a pair of principals u and v, we say that F is* positively monotonic to a single edge variable *with regard to u and v if for every such G and for every pair of edge weights w and w' we have that $F(G(w), u, v) \geq F(G(w'), u, v)$ whenever $w \geq w'$.*

Negative monotonicity can be similarly defined. It is not hard to see that if a trust function $F(G, u, v)$ is monotonic to an edge variable (either positively or negatively), then we can easily compute the minimum of $F(G, u, v)$ by simply evaluating F over extensions of G in which the missing edge variable is assigned the maximum and minimum possible weights.

This property can be similarly applied to the case of multiple undefined edges. Specifically, if F is monotonic to each edge variable with regard to u and v, the bounds of $F(G, u, v)$ can be straightforwardly obtained by taking the minimum and maximum of each variable respectively.

Given the above observation, the question of trust bounding can thus be answered easily if we can establish the monotonicity property of a trust function regarding certain edge variables. To distinguish the roles of different edges, we further define direct edges and indirect edges.

Definition 7 (Direct and Indirect Edges). *Consider a trust graph $G = \langle P, E \rangle$, a source principal u, a target principal v, and an edge $e = (x, y, w)$. The edge e is said to be a* direct edge *with respect to the target principal v if and only if $y = v$. Otherwise, e is said to be an* indirect edge *with respect to the target principal v.*

Next we examine the monotonicity of existing trust functions regarding these two types of edges.

5. ANALYSIS RESULTS

5.1 Monotonicity: Direct Edge Weights

We first focus on the case of direct edge variables. We argue that a well-designed trust function should be positively monotonic to a direct edge variable. To understand the intuition behind this assertion, suppose we want to bound $F(G, u, v)$. A direct edge variable $w_{s,v}$ represents the entity s's opinion of v. Intuitively, positive opinions should improve an individuals overall reputation, although the absolute degree of improvement may depend on the reputation of the individuals contributing opinions. However, a *positive* opinion should not cause a reputation to *decrease*. One may wonder whether negative trust inferences may happen: i.e., Alice distrusts Bob so much that if Bob thinks highly of a person, Alice will take the opposite opinion. Although this may happen in real human communities, this behavior is dangerous in online decentralized systems, as it opens the door to easy trust manipulations. That is, although it would be hard for an attacker to be trusted by everybody, it would be very easy for the attacker to be distrusted by most entities. If we allow negative trust inference, such an attacker may (easily) accrue a bad reputation with the sole purpose

of positively rating trusted services in an effort to decrease the overall reputation of that service. Positive monotonicity prevents this type of attack.

Recommendation Based Functions. It is not hard to show that this class of functions is monotonic with respect to a direct edge weight $w_{u,v}$. The essential reason is due to the observation that was mentioned previously: a direct edge weight only contributes to the local trust of a witness u for some target entity v, but not to the weight of the witness when her opinion is considered by others. Thus, when the value of a direct edge variable increases, the weighted trust can only change in one direction, depending on the weight on the witness u. In most recommendation-based trust functions, weights are nonnegative. Therefore, they are all positively monotonic to a direct edge variable, which is consistent with our argument above.

The only exception is Credence, whose weights on witnesses' local trust can be in the range $[-1, 1]$. Therefore, Credence could be negatively monotonic to $w_{u,v}$ if the weight of u is negative (i.e., u's voting records are negatively correlated with that of the source of the trust evaluation). As discussed above, this property allows an attacker to manipulate through negative trust inferences. Specifically, it could vote negatively on well-known high quality items, causing a strong negative correlation between itself and the source of a trust evaluation, so that its vote on the target entity or item would have a much higher impact on the final trust. One may wonder whether the effect would be the same when the attacker tries to form a strong positive correlation with the source. The different is that by voting negatively on well-known high quality items, an attacker can have a strong negative correlations with multiple honest entities. Thus, one manipulation of his voting history can be used to attack multiple entities. On the other hand, form a strong positive correlation with one entity usually does not result in the same with another honest entity, thus limiting the manipulation power of an attacker.

Topology Based Functions. This class of trust functions is also monotonic regarding direct missing information. A direct edge weight $w_{u,v}$ only positively monotonically affects the strength of a path, but not the weight of the path. Since the common functions to combine the strengths of multiple paths (e.g., MAX, AVG) are all monotonic to the strengths of individual paths, we can easily see that $w_{u,v}$'s impact to the final trust score is also positively monotonic.

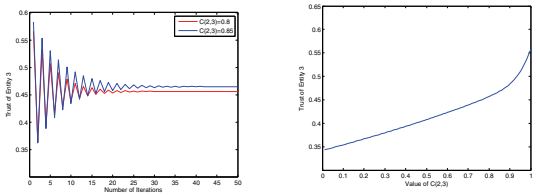
Take NICE as an example. If an entity a wants to evaluate the trust of another entity v, it first computes the strength of the possible (disjoint) paths from a to v. Obviously, the edge weight $w_{u,v}$ will only affect the strength of one specific path p. And since $u \neq a$ (otherwise, a has direct experience with v and can use its local trust directly), $w_{u,v}$ would not affect p's weight. (In NICE, the weight of a path is determined by the weight of its first hop.) Since the weight of a path is nonnegative, the trust function is positively monotonic to a direct edge variable.

Matrix Based Functions. Matrix based trust functions are more complex to analyze than the prior two classes of trust functions. Due to their usage of specific iteration operations, it is difficult to provide a general proof of monotonicity. Therefore, we chose three representative matrix based trust functions—Path Algebra, PeerTrust, and EigenTrust—and make a case-by-case analysis.

The positive monotonicity of Path Algebra regarding direct edge variables is relatively easy to establish. Note that all the elements in the matrix are nonnegative, and both of the concatenation and the aggregation functions described in [9] are positively monotonic. For each iteration, the matrix operation can be viewed as a composition of the concatenation and aggregation function. This implies that any feasible combination of possible such functions gives a positively monotonic trust function. We omit the formal proof due to its simplicity.

PeerTrust is more complicated to analyze than Path Algebra, since subtraction is used during the computation. As such, the transient trustworthiness of the entities may fluctuate between iterations. However, after the computation converges, a positive monotonicity property can be observed. A full proof of PeerTrust's monotonicity exists in the full version of this paper.

EigenTrust is one of the most influential trust functions in the literature. Although the general principle of EigenTrust is similar to that of PeerTrust, EigenTrust involves a normalization process that makes its analysis much more challenging. Specifically, EigenTrust is based on a stochastic matrix derived from the feedbacks in a system, such that the sum of each row must be 1. Therefore, we cannot apply the approach used to prove the monotonicity of PeerTrust to analyze EigenTrust, because standard matrix inverse does not existing for a stochastic matrix. To date, we have been unable to establish a proof of the monotonicity for EigenTrust. However, we cannot find a counterexample either. We have conducted extensive simulations over various reputation system settings, and all the results suggest that EigenTrust is positively monotonic to direct edge variables. Figure 2(b) shows how the final trust score of one entity changes when varying a direct edge weight from 0 to 1 in a small decentralized system. Figure 2(a) compares the temporary trust score of this after each iteration of the algorithm.



(a) The change of trust score after each iteration (b) Monotonicity shape

Figure 2: Simulation observation on EigenTrust

5.2 Monotonicity: Indirect Edge Weights

Recommendation Based Functions. Recommendation based functions are not always monotonic regarding indirect edge weights. Whether monotonicity holds mainly depends on how weights are computed. For example, Credence computes the weight of a witness based on the correlation between historical ratings of the source of trust evaluation and those of the witness. In this context, a witness sharing similar votes with the source is assigned a higher weight in the

aggregation of local trust. For this trust function, monotonicity is not guaranteed. For instance, consider the case where the rating on an entity v by the source u is 0.5. If a witness s has a rating for v that is close to 0.5, it achieves a high weight. However, a lower rating like 0.1 or a higher rating like 0.8 for v would lower the weight assigned to s, causing a non-monotonic change of the final trust score.

Compared to *Credence*, MDNT [11] computes the weight of a witness based on its historically correction record. After each transaction, the source entity of the evaluation will compare the quality of the service with the recommendations gathered before the transaction. A witness will get a positive feedback if his recommendation is matches the source entity's experience, and a negative feedback otherwise. After a period of time, every recommender establishes a history about the accuracy of his recommendations. Witnesses with higher correction rates will be assigned higher weights. For this trust function, monotonicity holds regarding an indirect edge variable, but the function may be negatively monotonic.

Note that even if a trust function is not monotonic to an edge variable, it does not mean that we cannot solve the trust bounding problem efficiently. It simply means that we need to study the specific design of such trust functions in a case-by-case manner. For example, even though Credence is not monotonic, given the voting history of the source of trust evaluation $(r_1, \ldots, r_n)$ and that of a witness $(r'_1, \ldots, x_i, \ldots, r'_n)$, it is not hard to compute when the correlation between these two vectors is minimized and maximized (i.e., when they have the same or opposite direction in the multi-dimensional space that two vectors are in).

Topology Based Functions. Topology based functions are not always monotonic regarding indirect missing information either. In contrast to a direct edge weight, an indirect edge weight w_e may affect both the strength of a path and the weight of the path; this may lead to a non-monotonic polynomial function of w_e. Below is a concrete example of the reputation system introduced in NICE [6].

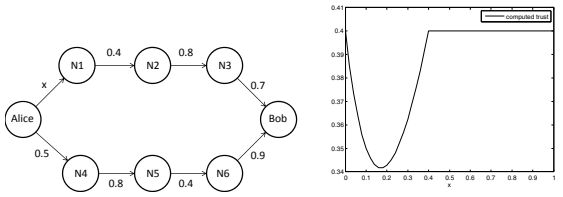


Figure 3: An example showing NICE is non-monotonic

As shown in figure 3(a), Alice wants to evaluate the trustworthiness of Bob. She first computes the strength of paths connecting to Bob with the weights of the edges in the paths. Then she uses *weighted sum of strongest disjoint paths* to merge the computed trust ratings. The two disjoint paths in the figure are $path_1 = \{Alice \to N_1 \to N_2 \to N_3 \to Bob\}$ and $path_2 = \{Alice \to N_4 \to N_5 \to N_6 \to Bob\}$. The weight on $Alice \to N_1$ is missing. Following the NICE algorithm, the strength of a path is the minimum weight of all its edges. And the weight of a path is the weight of its first hop from Alice. Therefore, we have the strength of $path_1$ as $\min\{x, 0.4\}$ and the strength of $path_2$ as 0.4. The weights of these two paths are x and 0.5, respectively. Then the final trust score of Bob from Alice's point of view can be expressed as in equation 5.2.

$$t_{Alice \to Bob} = \begin{cases} \frac{x^2+0.2}{x+0.5} & \text{when } x \leq 0.4 \\ \frac{0.4x+0.2}{x+0.5} & \text{when } x > 0.4 \end{cases} \quad (8)$$

The parabola-shape curve in Figure 3(b) indicates that this trust function is not monotonic.

Table 1: A summary of the monotonicity of our surveyed trust functions

Class	Function	Direct	Indirect
Recommendation	MDNT	Yes	Yes
	Credence	No	No
Topology	NICE	Yes	No
	Beta	Yes	Yes
Matrix	Path Algebra	Yes	Yes
	PeerTrust	Yes	Yes
	EigenTrust	Uncertain	No

Matrix Based Functions. As for Matrix based functions, we can show that the trust functions defined by Path Algebra are also positively monotonic to indirect edge variables. We can also derive a rigorous proof of monotonicity with respect to indirect edge variables for the PeerTrust trust function. For the details of the proof please refer to the full paper. However, for the EigenTrust trust function, no monotonic cases happened based upon our simulation results.

Table 1 summarizes the observations that we have made regarding the monotonicity of our candidate trust functions with respect to direct and indirect missing information.

6. RELATED WORK

Quite a few trust functions and reputation systems have been proposed in the literature over the last decade; for excellent surveys of these results, see [2, 4]. However, to the best of our knowledge, our work is the first to study the trust evaluation when certain information is missing in a reputation system. Next, we briefly overview recent research on securing reputation-based trust management.

Much work has been done specifically to prevent attacks in P2P systems. Zhang et al. [14] suggests that trust functions are effective when assuming that the service providers behave consistently, i.e., they always try their best to provide good services. However, a reputation system can be easily manipulated when an attacker adapts its behavior to take advantage of a trust function. Therefore, the authors propose a statistics based algorithm to test whether the transaction history of a service provider is compatible with that of an honest entity. They also develop a scheme that combines the above test with trust functions to force an attacker to behave relatively consistently, and thus increasing the cost of manipulation. In [1], Damiani et al. identify three attacks specific to reputation based systems, namely Pseudospoofing, ID Stealth and Shilling. The authors design a protocol called *XRep* which defends against all of the above attacks.

Many other works investigate defences against manipulation and other adversary activities in reputation systems

with a more general system setting. For example, CORE [8] only takes positive values of indirect ratings to prevent denial of service attacks that use malicious negative feedbacks against reliable entities. *TrustGuard* [10] combats malicious oscillatory behaviors with a strategic oscillation guard based on Proportional-Integral-Derivative controller. The system confines feedback to unforgeable transaction proofs and separates out dishonest feedbacks with a similarity measure to rate the reliability of a feedback. To prevent malicious entities subverting the system by reporting false trust values, a secure EigenTrust is implemented with the help of distributed hash table. It asks multiple entities to compute the trust value for an entity and then pick up the correct trust from non-malicious entities and eliminate the conflicts arising from malicious entities by a majority vote. In NICE, cookies are assigned to infer the trust of the target entity. However, a malicious entity may discard the cookie with a low value to hide his past misbehavior. Lee et al. propose to use negative cookies, which are feedbacks with low ratings. These cookies are stored by the issuer instead of by the target so that they cannot be discarded by the target.

7. CONCLUSION AND FUTURE WORK

Due to the decentralized nature of reputation systems, it is common that only a subset of feedback reports might be available at the time of a given trust function invocation. As such, it is important to examine the *quality* of reputation scores that can be obtained using incomplete information. In this paper, we initiate a first effort to investigate mechanisms for quantifying the effects of missing information. We formally define the trust bounding problem, which sets out to derive the minimum and maximum trust score of an entity when considering all possible values of the missing information. We identify two different types of missing information, and study the monotonicity properties of representative trust functions from the literature. Our investigation shows that most existing trust functions are monotonic with respect to *direct* missing information for the target of a trust evaluation, and thus the trust bounding problem can be easily solved in that context. On the other hand, the impact of *indirect* missing information on trust functions is more complicated. We have identified trust functions that are monotonic to indirect missing information, as well as some other functions that are not.

This work represents only a first step toward dealing with missing information in reputation systems. Many interesting avenues of follow-up work still remain to be explored. For instance, EigenTrust is one of the most well-known and influential trust functions in the literature, yet we cannot establish any formal proofs of its monotonicity regarding direct missing information due to its unique normalization process. One promising approach would be to leverage group inverses [7] as a mathematical tool to reveal the connection between the entries in an approximately inverse matrix and a variable in the original matrix. Formally establishing the monotonicity of EigenTrust would help better understand the properties of other matrix-based trust functions. Another interesting direction is to consider a probabilistic interpretation of the reputation quality problem (as mentioned in Section 1), and investigate various sampling techniques to accurately and efficiently estimate an entity's trustworthiness. This could lead to reliable distributed algorithms for reputation sampling in distributed systems. The samples collected could be used as input to existing trust functions, thereby producing reliable assessment metrics from unreliable primitives.

8. REFERENCES

[1] E. Damiani, S. D. Capitani, S. Paraboschi, F. Violante, and D. E. I. Politecnico. A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks. In *Proceedings of the 9th ACM CCS*, pages 207–216, 2002.

[2] K. Hoffman, D. Zage, and C. Nita-Rotaru. A survey of attack and defense techniques for reputation systems. In *ACM Computing Surveys*, volume 42, pages 1–31, Dec. 2009.

[3] A. Jø sang and R. Ismail. The Beta Reputation System. In *Proceedings of the 15th Bled eCommerce Conference*, 2002.

[4] A. Jø sang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. In *Decision Support Systems*, volume 43, pages 618–644, Mar. 2007.

[5] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *WWW Conference Series*, pages 640–651, 2003.

[6] S. Lee, R. Sherwood, and B. Bhattacharjee. Cooperative peer groups in NICE. In *Proceeding of the INFOCOM 2003*, 2003.

[7] C. D. Meyer. The role of the group generalized inverse in the theory of finite markov chains*. *Review Literature And Arts Of The Americas*, 17(3), 1975.

[8] P. Michiardi and R. Molva. CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Proceedings of the 6th CMS*, pages 107–121, Deventer, The Netherlands, 2002.

[9] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *Proceeding of the ISWC*, 2003.

[10] M. Srivatsa, L. Xiong, and L. Liu. TrustGuard: Countering vulnerabilities in reputation management for decentralized overlay networks. In *Proceedings of the 14th WWW conference*, 2005.

[11] E. S. Staab. The Pudding of Trust Editor's Perspective. *IEEE Intelligent Systems*, 3(4):19(5):74–88, 2004.

[12] K. Walsh and E. Sirer. Experience with an object reputation system for peer-to-peer filesharing. In *Proceedings of the 3rd NCDI conference*, 2006.

[13] L. Xiong and L. Liu. Building trust in decentralized peer-to-peer electronic communities. *Electronic Commerce Research*, 2002.

[14] Q. Zhang, W. Wei, and T. Yu. On the modeling of honest players in reputation systems. *Journal of Computer Science and Technology*, 24:808–819, 2009.

[15] H. Zhao and X. Li. VectorTrust: trust vector aggregation scheme for trust management in peer-to-peer networks. *The Journal of Supercomputing*, pages 1–25, 2011.