Technical Report

TR-2005-017

Countering Sparsity and Vulnerabilities in Reputation Systems

by

Ling Liu, Li Xiong

Mathematics and Computer Science

EMORY UNIVERSITY

Countering Sparsity and Vulnerabilities in Reputation Systems

Li Xiong
Department of Mathematics and Computer Science
Emory University
lxiong@mathcs.emory.edu

Ling Liu, Mustaque Ahamad
College of Computing
Georgia Institute of Technology
{lingliu, mustaq}@cc.gatech.edu

Abstract

While web applications provide enormous opportunities, they also present potential threats and risks due to a lack of trust among users. Reputation systems provide a promising way for building trust through social control by harnessing the community feedback in the form of feedback. However, reputation systems also introduce vulnerabilities due to potential manipulations by dishonest or malicious players. In this paper, we focus on important feedback aggregation related vulnerabilities, in particular, feedback sparsity with potential feedback manipulations, and develop resilient models and techniques. We propose similarity measures for differentiating dishonest feedbacks from honest ones and propose an inference framework to address the sparsity issues. We perform extensive evaluations of various algorithmic component of the framework and evaluate their effectiveness in countering feedback sparsity.

1 Introduction

Reputation systems have been shown to provide attractive techniques to address the risks and facilitate trust in a variety of web applications. Many electronic markets and online communities have reputation systems built in, such as eBay, Amazon, Yahoo! Auction, Edeal, Slashdot, Entrepreneur. A number of reputation systems have been proposed recently for web-based peer-to-peer applications, most notably, peer-to-peer file sharing systems [1, 8, 15, 28, 17, 33]. Trust management has been also proposed to facilitate trust in semantic web and to combat web spam [19, 23, 12]. By harnessing the community knowledge in the form of feedback, reputation based trust systems help participants decide who (what) to trust, encourage trustworthy behavior, and deter dishonest participation [22]. However, these systems also introduce vulnerabilities due to potential manipulations and dynamic behaviors of malicious peers. Among these, an important problem that has caught less attention so far is the sparsity of the feedback with potential manipulations of the feedback.

Feedback Sparsity. Reputation systems rely on historical transaction and feedback data to derive a reputation trust score for entities. In large scale peer-to-peer networks, each peer may have interacted with only a small number (percentage) of peers and hence has few feedbacks about and from other peers. It is been shown that limited reputation sharing or no reputation sharing when personal experiences are available [17] helps minimize the adverse effect of feedback attacks by malicious peers. However, when the feedback data is sparse, it is often not possible for peers to have personal experiences with each other. As a result, the problem of feedback sparsity has a major negative impact in reputation systems. When the feedback matrix is sparse, the system cannot evaluate the trust value most peers or the output evaluations suffer from accuracy.

Feedback Manipulation. A piece of feedback is simply a statement from a user about another user, typically, the service consumer about the service provider in a peer-to-peer network. There is no mechanism to guarantee that the statement is honest. Malicious users may manipulate the feedback data in a way in order to benefit themselves or damage the system. A main technique in the reputation systems proposed so far is to associate a credibility weight with each peer (as versus the service reliability of the peer). Theoretically, a peer who has consistently provided honest feedback will be associated with a higher credibility weight so the system can differentiate honest feedback from dishonest ones. It has been shown [28] that the user similarity based on previous feedbacks about common peers can be used as a personalized credibility measure in the trust metric and is very effective against certain attacks including collusion among a group of peers who provide good ratings within the group and bad ratings outside the group. Unfortunately, when the feedback is sparse, it is highly probable that two users have not interacted with any common set of peers and thus the feedback based similarity between two users cannot be computed.

Interestingly, the problem of feedback sparsity and manipulation are closely related. Feedback sparsity concerns with the quantity of feedback data. Feed-

back manipulation concerns with the quality of feedback data and in most cases may affect the quantity of good quality feedback data. The two problems also have a magnifying effect on each other. On one hand, when adversaries pollute the feedback data, the valid or usable feedback data becomes sparse, which makes sparsity problem worse. On the other hand, when feedback data is sparse, the feedback related attacks may be magnified and have a more detrimental effect.

Research in trust inference [30, 4, 27, 11] addresses trust propagation of initial trust relationships assuming certain transitivity of trust. They differ with reputation systems in that they assume the initial trust relationship is predefined among nodes and the proposed approach typically uses graph theoretic models or matrix operations to propagate the initial trust relationships. Some recent works adopted certain inference techniques in reputation systems. A recent notable work EigenTrust [15] derives initial trust based on personal feedback and perform a global trust propagation till it finds the Eigenvector of the initial trust matrix. However, it has been shown that the model is vulnerable to certain types of attacks where adversaries exploit the trust transitivity property by creating false trust links.

While these works shed lights on the potential adoption of inference techniques to address sparsity issues in reputation systems, a few research challenges remain. First, how do we derive the initial relationships among nodes and what information or relationships do we use to perform inference so that it is robust to potential user manipulation? Second, how do we design the inference model and select the right parameters such as depth of inference and inference functions to deal with sparse data while on the other hand to avoid unnecessary cost. Finally and importantly, how does the inference model cope with potential feedback manipulations in the reputation system?

Bearing these questions in mind, we propose a similarity based framework and study experimentally how different inference parameters and models perform in countering sparsity and vulnerabilities. The paper has a few unique contributions. First, we propose a general algorithmic framework for reputation computation and classify existing reputation schemes according to the framework. Second, we focus on one set of reputation algorithms that we refer to as neighborhood based schemes and formalize the sparsity problem and explore different attack strategies related to feedback vulnerabilities. Third, we study how variant algorithmic component of the above reputation schemes help coping with feedback sparsity and vulnerabilities.

The rest of the paper is organized as follows. Section 2 defines certain terms and examines the problem space by defining threat models we consider and the sparsity problem. Section 3 presents a similarity inference scheme with variant algorithmic techniques. Section 4 experimentally studies how the different algorithmic components help coping with the feedback sparsity and vulnerabilities. Finally, Section 6 summarizes the paper and discusses a few directions for future research.

2 Problem Statement

We first define certain terms that will facilitate our discussion and comparison of various reputation schemes. We then examine the problem space for feedback sparsity as well as feedback integrity by defining the threat model and attack strategies.

2.1 Terms and Definitions

Peer-to-peer community. The peer-to-peer community consists of N peers who perform transactions with each other. The community can be built on top of a client server network or a pure P2P network. It can be also generalized into an online community that consists of N users and M service providers. In P2P community, the number of users and service providers happen to be the same as each peer serve both as a client and a server.

Transaction. The community is defined by interactions or transactions among peers. These transactions may include downloading files, storing data, or monetary transactions. Each transaction has a client (service consumer) and a server (service provider) and each peer may serve as a client in one transaction and as a server in another.

Transaction feedback. A transaction feedback is a statement issued by the client about the quality of a service provided by the server in a single transaction. This can be collected explicitly or derived implicitly.

Personal feedback. A personal feedback or opinion is a user's general impression about a service provider based on its personal experiences with the server. It can be derived from its feedback on all the transactions that are conducted with the server in the past, e.g. as the percentage of positive transactions or the average rating for each transaction. We can represent all the personal feedback in the network as a user-user opinion matrix, where each cell represents an opinion from a given user about a given server.

Reputation Trust. The goal of a reputation system is to compute a reputation trust for a given service provider from a given user's perspective. We refer to the evaluating user as *source* and the user to be evaluated as *target*. A reputation trust matrix is a user-user matrix, where each cell represents a reputation trust

score for a given service provider from a given user's perspective. We call the reputation trust global if the reputation trust for each service provider is the same across all the users. Otherwise, it is personalized.

In some trust schemes, if a user u has direct interactions with a server s, then u's rating of s is treated as s's trustworthiness from the u's point of view. In PeerTrust as well as some other schemes, u still needs to consider the other users' ratings about s when evaluating s' trustworthiness. A peer's trustworthiness is the combination of the user's personal opinion and the community feedback. Thus we differentiate the user rating from trustworthiness.

Credibility. Credibility of a user u indicates how credible u is in providing feedback or rating is in general. In contrast, trustworthiness indicates how reliable u is in providing service. Some works use trustworthiness as a general notion. In PeerTrust, we argue that credibility should be differentiated from trustworthiness (reliability). User a may trust user b for its ratings or recommendations but not necessarily its service. It is also referred to as recommendation trust in some literature [4] as versus service trust. We will use the term credibility for this purpose and reserve the term trust for the reputation based service trust (reliability) in this paper. Credibility of a peer can be collected or defined explicitly or derived computationally.

2.2 Threat Model

Reputation systems have to cope with potential manipulations from adversaries. A common goal for malicious users is to boost their own ratings or decrease the ratings of other peers by manipulating feedbacks so they can perform malicious services when other peers select to perform a transaction with them. A feedback manipulation attack corresponds to the addition of noise to the feedback data (training data). A common strategy for adversaries is to provide false ratings. We consider various dimensions of these types of attacks in this section. There are other different attacks including strategically milking a reputation or oscillating behaviors [17, 26] and other different attack incentives including free riding [2, 21, 32]. They are not considered in this paper and interested readers may refer to the references.

Attack Goals. Based on different goals of the attacks, we consider the following two types of attacks.

- *Random attacks*. The goal of this type of attacks is to reduce the overall performance of a system as a whole. They are not directed at any particular users.
- *Target attacks*. These are attacks that try to force the ratings for a target user to a particular target value. For example, in a nuke attack, the goal is to force all predicted ratings of targeted users to the minimum rating. Similarly, in a push attack, the goal is to force the ratings to the maximum rating.

Attack Models. We also consider two different types of attacks models.

- *Non-Collusive*. In the non-collusive model, individual malicious users do not know each other and they each do something bad and hoping it will affect the system.
- *Collusive*. In the collusive model, multiple malicious peers may form a group and collude with each other in order to achieve their goal. The goal is typically targeted towards boosting the ratings of the whole or part of the group.

  A related attack is for an individual user to create multiple fake profiles and act as a collusive group. We will treat this same as the collusive attack. We do assume adversaries have to pay a cost to create a profile so it is not feasible to create infinite false profiles. The goal of designing a robust algorithm is to maximize the noise level it can tolerate so an adversary has to pay a high cost in order to achieve their malicious goal.

Attack Strategies. Some attacks may be specifically designed to exploit a particular weakness in a specific algorithm or class of algorithms. For example, for systems that does not differentiate trust (reliability) and credibility, one simple strategy for the malicious group is to have part of the peers act as front peers or moles [18, 10]. These peers always cooperate with other peers in order to increase their reputation and then provide misinformation to promote other malicious peers. We will discuss them in more detail when we review different reputation systems in Section 5.

Differentiating trust (reliability) and credibility helps systems to avoid the front peer attacks and are more robust to dishonest feedbacks. It has been shown [28] that using user similarity as a credibility weight when aggregating community feedback provides promising results in defending against dishonest feedback attacks in a non-collusive model and a naive collusive model.

So in the rest of the paper, we will focus on the similarity based models and explore more sophisticated attack strategies and the sparsity problem and develop enhanced framework to defend against them.

As malicious users do not collude with each other in non-collusive setting and the individual attack strat-

egy is straightforward, we focus on the attack strategies in the collusive model. Assuming the goal for the group is to boost the ratings for the group. There are different strategies that the users may use in rating other peers that are outside their group. Similar to the shill attack designs from [16] in a collaborative filtering context, we study the following strategies.

- *Collusive.* A straightforward way is for each collusive user to rate each other in the group with the maximum rating and rate peers outside the group with a minimum rating. The hope is while they boost their own ratings, they also decrease other peers' ratings or damage the performance of the reputation system. However, by doing this, they may end up with having very low similarity to the normal users and in turn their ratings will not be counted as much.

- *Collusive Camouflage.* A more sophisticated attack is for the malicious users to rate each other within the group with maximum rating but rate peers outside the group honestly so that they will be similar to more existing honest users, and thus, have a larger effect on boosting their own ratings. Hypothetically, this strategy will allow adversaries to amount a more detrimental attack and boost their ratings by camouflaging as honest users.

2.3 Sparsity Problem

Now we consider the sparsity problem for using such a similarity based trust scheme such as the one used in PeerTrust [28]. We compute the reputation trust of a user as a weighted average of previous ratings about the user. The credibility weight is a personalized similarity measure between the evaluating peer and the peer who provides ratings. Concretely, peer w will use a personalized similarity between itself and another peer v to weight the feedback by v on any other peers. We define the similarity by modeling the feedback by v and the feedback by w over a common set of peers for which both v and w have rated as two vectors and computing the similarity between the two feedback vectors.

It is important to note that the similarity is defined over the common set of peers that both peers have rated. When the input ratings matrix is sparse, two given peers may have a very limited number or zero number of co-rated peers and the similarity can not be derived. As a result, the reputation trust can not be computed.

Now we formally analyze what is the probability of an undefined prediction given a sparse feedback matrix. Given a source user a and a target service provider j, let U_j denote the set of users who have rated j, let N_a denote the set of neighbors of a. N_a is selected by the neighborhood selection technique out of all the users who have co-rated items with a. We can compute the reputation trust by aggregating the feedback from those neighbors of a who have rated j, computed as the intersection between U_j and N_a.

When the input matrix is sparse, it has three effects. First, a will have a limited number of neighbors, i.e. N_a is small, as there is less chance that other users will have co-rated items with a. Second, j will have a limited number of ratings, i.e. U_j is small. As a result, the probability of the intersection of N_a and U_j being empty will be high which results in an undefined prediction.

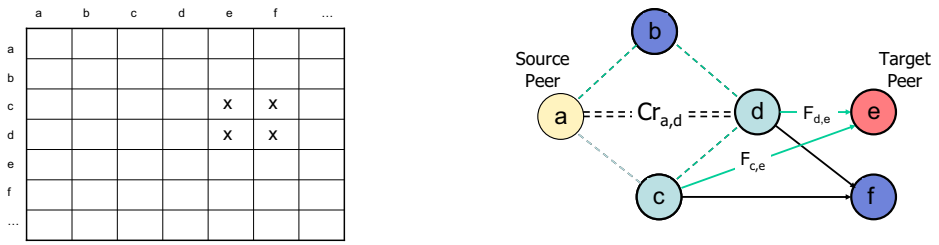
Suppose each user rates r users among the other $n-1$ users. The average number of ratings per user is also r. For simplicity, we assume all users who have co-rated items with a are considered as as neighbors and no neighborhood selection is used. In this case, the probability of a user not belonging to a's neighborhood is the probability that it does not have co-rated item with a. The probability of an undefined prediction is the probability that all of the users who have rated item j do not belong to a's neighborhood. So the final probability is given by Equation 1.

$$p(n,r) = \left(\frac{\binom{n-r}{r}}{\binom{n}{r}}\right)^r \tag{1}$$

In a large open community, most users will only interact with a small set of users and may provide ratings to an even smaller set of users among those. The general sparsity problem refers to such situation where while n, the number of users, could be of the order of thousand, r, the number of ratings per user, could be very small. The resulting probability of undefined prediction can be high. When a new user joins the community, it does not have many ratings about other users nor does it have many ratings from other users. Suppose a new user a only has r_a ratings ($r_a \ll r$). The probability of undefined prediction becomes $\left(\frac{\binom{n-r_a}{r}}{\binom{n}{r}}\right)^r$. Even though the input ratings matrix may not be sparse in general, the probability is still fairly high due to a small r_a. This is sometimes referred to as cold start problem.

3 A Similarity Inference Framework

In this section, we propose a framework that uses similarity as credibility weight and an inference scheme based on similarity weight to provide sparsity resilience and robustness.



(a) Personal Rating Matrix

(b) Rating and Neighborhood Graph

Figure 1: Similarity Inference Scheme: Illustration

3.1 Overview

We first model our problem in a graph theoretic way. Recall that in our personal rating matrix, each cell represents a personal rating from a node about another node. Suppose each peer is represented as a node in the graph. If peer u has interacted with and has a personal rating about peer v, it corresponds to a directed link from u to v. Each peer then has a set of outgoing links representing the ratings it has given to other peers and a set of incoming links representing the ratings it has received from other peers. Figure 1 illustrate a partial personal rating matrix and a corresponding graph with solid links representing the personal rating links.

When peer u is evaluating the trustworthiness of peer v, it can use its personal rating if it is available (direct link from u to v) or the community reputation based on the community ratings about peer v (all incoming links to v). Without loss of generality, in our framework, the trust value is defined by a linear combination of the user's personal rating and the community reputation. The user can tune the weight if it desires to give more weight to its personal experiences when it is available. In the rest of the chapter, we will focus on the computation of the community reputation.

The community reputation aggregates all the ratings about v. As we have discussed earlier, a common strategy to combat the dishonest feedback is to use certain metric to weigh the ratings during aggregation. There have been a few classes of metrics that have been used. One common way is to use the peer's previously determined reputation score to weigh its feedback. However, as we have discussed earlier, a user may have personal experiences with another user and get opinions on the user for performing service but it does not indicate any opinions on the user for providing feedback. It has been shown these schemes are vulnerable to certain types of attacks where adversaries exploit the transitivity property of the trust links and create fake links.

We argue that we need to derive a separate credibility link between users so they can use them to weigh each other's ratings. Intuitively, a user is more willing to trust another user v's ratings if v has given similar ratings to the same set of peers that u has done before. If we treat all the ratings each peer has given to others, we can derive certain relationship between users in terms of their feedback filing and use this as a credibility measure. As a result, in addition to the personal rating links, we add another type of links, credibility links, with each link representing the similarity between a pair of users. The credibility links are illustrated as dashed links in Figure 1.

For some pair of users, the similarity can be directly derived using certain similarity measure between the ratings they have given before over a common set of peers. And we will discuss in detail different similarity measures and study experimentally how they affect the performance. However, when the input data is sparse, it is not always possible to derive the direct similarity or the derived link may be unreliable because they are based on too little information.

This motivates us to explore the transitive associations of similarity. If a user u is similar to v, v is similar to w, u should be somewhat similar to w. In other words, similarity may propagate through the network. The indirect similarity link is represented as double dashed link in Figure 1. This is particularly

important when the input feedback matrix is sparse. We will present algorithms that help peers find indirect neighbors and compute indirect similarity weight for indirect neighbors.

In summary, the framework uses user feedback similarity to weigh their ratings and uses similarity inference to compute the weight for indirect neighbors. It has the following main steps.

- Compute initial trust based on personal experience. The result of this is the personal experience matrix.
- Compute direct similarity based on the ratings matrix.
- Find indirect neighbors and compute indirect credibility.
- Aggregate user feedback (possibly from a selected neighborhood of users) based on the credibility of the users.

There are a number of research questions we would like to study in the proposed scheme. First, what kind of similarity do we use? Second, what kind of inference model do we use and how do we determine the depth of the inference and the inference functions? Third, do we use all the user's neighbors or use a subset of the neighbors and what criteria? Finally, how do all the algorithm variants handle the sparsity issue as well as the attacks?

In the rest of the section, we present the details of each component of the scheme and study these questions by examining various algorithms and techniques and their effects on countering sparsity and vulnerabilities.

3.2 Computing Credibility using Similarity

We first derive the weight of direct links between two users. Assuming the personal feedback $v_{a,i}$ is derived as an average of all the transaction feedback (satisfaction) from a to i in previous transactions. Complex measures can be also used to handle strategic oscillating behaviors of nodes [26]. If we treat all the ratings each peer has provided in the past about other peers as a vector or variable, we can compute similarity between two users using different measures. The intuition is that a user will trust another user's ratings if the latter has given similar ratings as the user does previously. Depending on different similarity measures, the direct similarity can be undefined. Several different similarity measures have been used in collaborative filtering systems. We adopt the following three measures in our framework.

- *Pearson correlation.* Pearson correlation is widely used in collaborative filtering systems. It computes the degree of linear relationship between two variables. It ranges from +1 to -1 where +1 means there is a perfect positive linear relationship. The Pearson correlation coefficient between user a and b is calculated as:

$$w_{a,b} = \frac{\sum_{i \in I_a \cap I_b}(v_{a,i} - \bar{v}_a)(v_{b,i} - \bar{v}_b)}{\sqrt{\sum_{i \in I_a \cap I_b}(v_{a,i} - \bar{v}_a)^2 (v_{b,i} - \bar{v}_b)^2}} \quad (2)$$

- *Vector cosine similarity.* The vector similarity measure is another widely used technique in which each user is represented by a vector of ratings in the $|I|-$ dimensional space, where I is the set of items. The similarity between two users a and b is calculated as the cosine of angle between the two corresponding vectors (normalized inner product):

$$w_{a,b} = \frac{\sum_{i \in I} v_{a,i} v_{b,i}}{\sqrt{\sum_{i \in I_a} v_{a,i}^2} \sqrt{\sum_{i \in I_b} v_{b,i}^2}} \quad (3)$$

- *Euclidean distance.* In our previous work [28], we have also used the root-mean-square or standard deviation (dissimilarity) of the two feedback vectors to compute the similarity.

$$w_{a,b} = 1 - \sqrt{\frac{\sum_{x \in I_a \cap I_b} (v_{a,i} - v_{b,i})^2}{|I_a \cap I_b|}} \quad (4)$$

An inherent difference between the Euclidean distance and the other two is that it does not perform a normalization of the feedback. Intuitively, normalization of feedback will minimize the effect of user bias. For instance, one user may always give high ratings for all other users while another may always give low ratings. When the feedback is normalized, it is the relative ratings that matter and these two users will have a high similarity despite their bias. On the other hand, the Euclidean distance will treat these two users as dissimilar.

Other potential measures include Spearman correlation which takes the ranks of the two variables and computes a ranked version of Pearson correlation. Existing collaborating filtering literature [5, 13] has reported that Pearson correlation and Spearman correlation yield comparable and slightly better accuracy than vector cosine similarity in recommender systems typically evaluated by movie rating datasets. We will study experimentally how these measures perform in our context with sparse feedback data and potential feedback manipulations in Section 4.

3.3 Similarity Inference

We refer to all the users who have a direct similarity weight defined with u as u's direct similarity neighbors. Once the direct similarity neighborhood is formed, when u needs to evaluate another service providers trustworthiness, it asks its neighbors for their opinions about the target peer and aggregate their opinions. (The user can either use all its neighbors or select a subset of neighbors. We will discuss neighborhood selection techniques later in this section). However, when the input data is sparse, each peer may have a very limited number of direct neighbors and thus the chance increases that none of its direct neighbors has an opinion about a given target peer. In this case, it is natural to consider the neighbors neighbors in order to get a larger neighborhood for future evaluations. In this section, we explore the transitive associations between similarity and present algorithms to find indirect neighbors of a given user and propagate the similarity weight between the user and indirect neighbors.

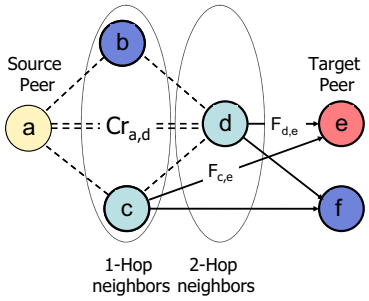


Figure 2: Discovering Indirect Neighbors: Illustration

Discovering Indirect Neighbors We can model the problem as a transitive closure problem to find all nodes that are reachable by node u by at least one path. (Note here that the path consists of only similarity links). Intuitively, a user may be only willing to take the opinions of other users that are within certain distance. An indirect neighbor that is too far away on a similarity path stops being useful even if each node is perfectly similar to their immediate neighbors on the path. So we bound the path by a maximum path length L meaning we only consider the indirect neighbors within L hops of the source user u. If a node is reachable by u by at least l hops, we call the node u's l-hop neighbor. u's 1-hop neighbors are also u's immediate neighbors. Figure 2 shows the 1-hop (immediate an illustration of the similarity based neighbors.

Since we have a maximum path length bound, we do not need a full transitive closure algorithm. Instead, for each node, similar to the single source transitive closure problem, we can use a standard breath first graph search algorithm to find all the indirect neighbors that are within L hops. Since the input data may be sparse and each user only has a limited number of immediate neighbors, we expect the algorithm to be efficient.

Computing Indirect Weights The similarity weight for u's immediate neighbors are the direct link weight. For the neighbors greater than 1-hop, we need to compute an indirect weight for them. For a l-hop neighbor v, we consider all the l-length paths from u to v and compute an indirect weight based on these paths.

Formally, for each path, we infer the indirect weight for this path as a product of the direct weight on each link as in Equation 5. We can also add in a decaying factor at each step such that the similarity decays at each step and essentially a path decays to 0 if it is too long. Since we have bounded the path length by a maximum length in finding neighbors, we set the decay factor as 1 in our implementations.

$$p^i_{a,j} = w_{a,u_1} * w_{u_1,u_2} * \ldots * w_{u_{l-1},u_l} * v_{u_l,j} \quad (5)$$

If there are multiple paths of length l from node u to node v, we need a function to select or combine the inferred values from each path into a single value. We consider three such functions: maximum, minimum, and average.

- *Maximum Value.* Maximum function assumes an optimistic behavior of the user and selects maximum similarity value of the multiple paths. This is consistent with performing a generalized *or* operation over [0,1] valued beliefs in fuzzy logic. User u will believe another user j to an extent with which at least one of its closer neighbors trust j.

$$s_{a,j} = \min_i p^i_{a,j} \quad (6)$$

- *Minimum Value.* Minimum function assumes a pessimistic behavior of the user and selects minimum similarity values of the multiple paths. This is consistent with performing a generalized *and* operation in fuzzy logic. User u will only believe another user j to an extent with which all its closer neighbors trust j.

$$s_{a,j} = \min_i p^i_{a,j} \quad (7)$$

- *Average Value.* Average function computes an average of the similarity values of the multiple paths. Intuitively, this may give us the best of maximum and minimum function. However, average function is not associative so it may be harder to implement using standard algorithms and we will discuss the implementation later in this section.

$$s_{a,j} = \frac{\sum_{i=1}^{n} p_{a,j}^{i}}{n} \tag{8}$$

Neighbor Selection Once the neighbors are discovered and their indirect weights are computed, some of them may turn out to have a very low similarity with the source user. It is been observed in collaborative filtering systems that the neighbors with low correlations will greatly affect the prediction accuracy [5]. So in addition to the nave method that considers all the neighbors, we also consider two neighborhood selection techniques in our scheme.

- *Threshold Based Selection.* A threshold based neighborhood selection scheme sets an absolute similarity threshold, where all neighbors with absolute similarity greater than the threshold are selected. It has been suggested that high correlates can be exceptionally more valuable than those with lower correlations [5]. Hypothetically, setting a high threshold limits the neighborhood to containing very highly similar users, but may also results in a small or even empty neighborhood so it cannot compute the similarity based reputation for many users.

- *k Nearest Neighbor (kNN) selection* An alternative technique is to pick the top k neighbors in terms of similarity for each user. This works in a similar way to k nearest neighbor (kNN) classification algorithm.

Note that although the similarity based distance is defined in a symmetric way, i.e. the similarity between node a and b is the same as that between node b and a, the neighborhood can be defined in an asymmetric way. Consider a scenario where node a is connected to many nodes with high similarity and to node b with low similarity while node b is connected to a few nodes including node a with low similarity. In this case, node a may filter out node b from its neighbor list by setting a fairly high threshold while node b may very likely keep node a as its neighbor because of its limited number of connected nodes. This notion of relative distance provides a nice notion of meta-distance into the neighborhood selection.

We will evaluate the different neighborhood selection techniques with varying parameters such as the threshold value in Section 4 in terms of their effects on sparsity resilience and robustness of the algorithms.

When there is a neighborhood selection scheme, the threshold parameters or the k value can be also built into the neighborhood discovery process so that the neighbors who have a low weight will be pruned early on and the algorithm will be more efficient.

Matrix Representation If we represent the graph by a Boolean adjacency matrix M, M^l will gives us all nodes that are reachable with a length l path. If we represent the graph by a similarity weight matrix with each cell representing the link weight, assuming the weight is normalized for each user, we can also use matrix production M^l to compute the indirect weight. It essentially corresponds to using production as the path concatenation function and using weighted average as the path combination function.

If we normalize the similarity weight by each user, we can also interpret the similarity inference problem as a stochastic transition process similar to the random walker model in PageRank [20] and EigenTrust [15]. Imagine the source user a is trying to find similar neighbors, at each step, it hops to a neighbor according to the current users distribution of similarity.

The difference between our approach and web of trust or EigenTrust is that user a stops when it reaches the maximum path length. In the global inference approach such as EigenTrust, a users personal beliefs are washed out by the infinite aggregation of other users beliefs and the resulting eigenvector is a global trust vector.

It is worth mentioning, however, if we normalize the similarity weight by each user, then the similarity value does not have the semantic meaning any more, it only gives a relative rank of the users in terms of similarity. So the threshold based algorithm will not work with normalized similarity but kNN based scheme will.

Implementation Algorithm We implement a breath first search based algorithm of the above conceptual model in our first prototype.

Algorithm 1 depicts a sketch of the algorithm that is executed at node a. Node a first initializes its neighbor by adding itself into the list. Then at iteration step l, it gets all the l-hop neighbors and compute their indirect weights. The algorithm terminates after it retrieves all the L-hop neighbors.

At each iteration step, node a go through its current neighbor list, and retrieve all the immediate neighbors of its current neighbors, if the node is not in us neighbors list yet, it is considered a new l-hop neighbor and being added into the $newNeighbors$ list. If there are multiple paths from a to c, it users the given

Algorithm 1 getNeighbors(a,L, f)

```
Input: a, L, f, Output: N_a
N_a.put((a,1))
l ← 1
while l ≤ l_max do
  newNeighbors = φ;
  for b ∈ N_a do
    IN_b = GetImmediateNeighbors(b)
    for c ∈ IN_b do
      w_{a,c} = w_{a,b} * w_{b,c}
      if c ∉ newNeighbors then
        newNeighbors.put (c, w_{a,c})
      else
        newNeighbors.update                    (c,
        f(newNeighbors.get(c), w_{a,c}))
      end if
    end for
  end for
  N_a.add(newNeighbors)
  l ⇐ l + 1
end while
```

combination function (Maximum, Minimum or Average) to update its weight. Note that Average can be implemented by using Sum and Count and we skip the algorithm details here.

We can also adopt spreading activation algorithms for certain implementations of the above model to achieve better efficiency. Spreading activation models have been first introduced in order to simulate human comprehension through semantic memory and have been later applied to many associative retrieval problems [7]. The similarity inference problem can be naturally modeled as an association retrieval problem and various spreading activation algorithms can be used for implementing the computation. For example, the branch and bound algorithm [6] can be adopted to implement the above model if the path combination functions are associative (out of the functions we considered, maximum and minimum are associative, while average is not).

4 Experimental Evaluations

We performed a set of simulations to study the performance of the framework under different sparsity levels and threat models.

4.1 Experimental Design

For each experiment, we vary the sparsity of the input feedback and evaluate the accuracy and coverage of different parameters to study the effect of sparse feedback.

Based on the algorithmic scheme described in Section 3, we study the performance of the system experimentally with the following varying parameters.

- *Similarity measure.* We study the three similarity measures used to compute the direct similarity weight: Pearson correlation, vector similarity, and Euclidean distance.
- *Maximum path length.* We vary the maximum path length that is used for exploring indirect neighbors from 1 to 4. When 1 is used, only direct neighbors are considered.
- *Path combination function.* We consider the three path combination functions used to compute the indirect similarity weight when there are multiple paths between two nodes: maximum, minimum, and average.
- *Threshold based neighbor selection.* We also vary the absolute threshold used for selecting a subset of neighbors to aggregate the feedback from 0 to 0.75. When 0 is used, it is the same as using all the neighbors.

When one parameter is varied, we set the other parameters with a default value. Table 1 summarizes the main experimental parameters with their default values as well as the community parameters that we used for the experiments.

We also perform attacks according to the attack strategies we described in Section 2.2 and evaluate the performance of the system with a low sparsity and varying sizes of attacks. Table 2 summarizes the behaviors of malicious peers in different attack models.

4.2 Evaluation Metrics

We examine different algorithmic variants of each component in the framework and evaluate their performance focusing on the sparsity resilience and robustness of the collaborative systems. We use the following two main metrics.

- *Accuracy.* An accuracy metric empirically measures how accurate a system in predicting the reputation scores as a measure of the trustworthiness of peers.

  We use statistical accuracy metrics that compare how reputation system's computed reputation values for a peer differ from the peer's assigned reliability value. Typical statistics metrics include Mean Absolute Error (MAE), Root Mean Squared Error and Correlation error. All the above metrics generally provided same conclusions, so we only report Root Mean Square Error.

Table 1: Countering Feedback Sparsity: Experiment Parameters

	Parameter	Description	Default
Community Setting	N	# of peers in the community	100
	s	density of feedback	5%
	k	% of malicious peers in the community	0
	$mrate$	% of transactions a malicious peer acts malicious	100%
Trust Computation		Similarity Measure	Euclidean
	L	Maximum Path Length	2
		Path combination function	Average
	Cr_{min}	Neighbor selection threshold	0
	$nExp$	# of experiments over which results are averaged	10

Table 2: Countering Feedback Sparsity: Attack Models

Attack Models	Service Quality	Feedback Quality
Non-Collusive	Low	Random
Collusive	Low	Target maximum rating (within group) Random (otherwise)
Collusive Camouflage	Low	Target maximum rating (within group) Honest (otherwise)

- *Coverage*. Coverage measures the percentage of a test dataset that the reputation system is able to compute reputation value for. It is worth noting that coverage and accuracy is at odds in some cases. For example, using neighborhood threshold selection may increase accuracy but decrease coverage. We will therefore focus on both improving accuracy and coverage in various sparsity levels and threat models.

4.3 Countering Sparsity

We first study how variant algorithmic components in our scheme handle the sparsity of the feedback.

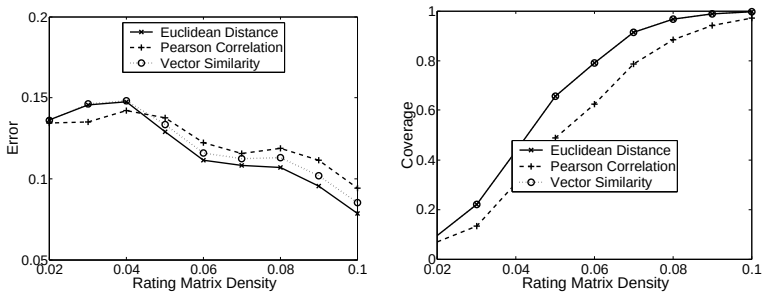
Similarity measures. There are a number of ways to derive the similarity measure as the credibility weight for aggregating the feedback of peers. We first study their effects on the sparsity reputation systems.

Figure 3 shows the trust computation error and coverage of using different similarity measures under different sparsity levels. The three similarity measures give comparable precision and coverage. In particular, Euclidean distance and vector similarity achieve better coverage than Pearson correlation as Pearson correlation can only be computed when two users have commonly rated peers.

Maximum Path Length. Now we study how the propagation of similarity affects the system in coping with sparsity. We first study the effect of the maximum path length that is used to explore for indirect neighbors.

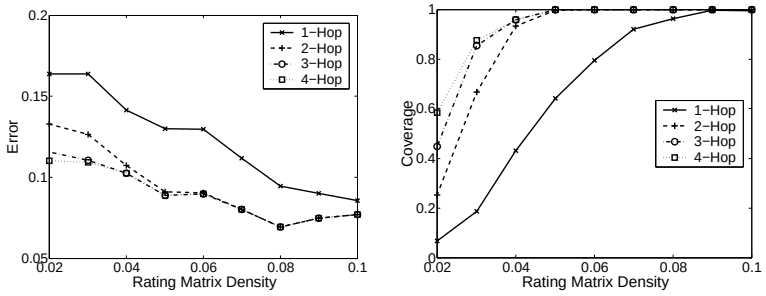
Figure 4 shows the trust computation error and coverage of using different maximum path length under different sparsity levels. We can make a few interesting observations. First, the 2-Hop algorithms provide significantly better precision and coverage than the basic 1-Hop algorithm (maximum path length = 1). The performance gain is more significant when the input data is extremely sparse. This shows that considering indirect neighbors is important in countering sparse data. It reflects the intuition that you need to talk to people outside your close circle to get the best information. Second, 3-Hop and 4-Hop algorithms do not perform much better than 2-Hop algorithm. It shows that considering neighbors that are far away does not help much in achieving a better precision and coverage in this case. It also reiterates the observation that limited reputation sharing or a local reputation scheme is desired to avoid the unnecessary computation costs. This can be potentially explained by the small world effect wherein most of the peers may be connected by length 2 paths in our scenario. It is worth noting however that a formal analytical study is needed to determine the optimal maximum path length for a given sparsity and distribution of the community.

Path Combination Function. We also study the



(a) Trust Computation Error (b) Trust Computation Coverage

Figure 3: Trust Computation Error and Coverage of Different Similarity Measure with Varying Sparsity Levels



(a) Trust Computation Error (b) Trust Computation Coverage

Figure 4: Trust Computation Error and Coverage of Varying Maximum Path Length with Varying Sparsity Levels

effect of different path combination functions when computing the similarity weight for indirect neighbors.

Figure 5 illustrates the trust computation error and coverage of the three path combination functions with varying sparsity levels. Interestingly, the maximum function has the lowest error rate with varying sparsity levels. This shows that by assuming an optimistic behavior (taking the maximum similarity value among multiple paths) peers can achieve better reputation accuracy given a sparse feedback input. Different path combination functions do not have any effect on the computation coverage.

Threshold Based Neighbor Selection. Finally we study the threshold based technique in selecting neighbors for trust evaluations.

Figure 6 illustrates the trust computation error and coverage of using different threshold values with varying sparsity levels. Interestingly, setting different threshold values does not have a significant impact on the trust precision. Having a threshold value of 0.75 increases the precision marginally but suffers a lower coverage.

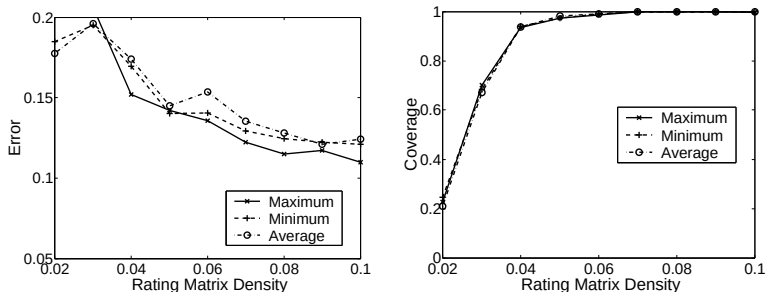
4.4 Effect of Feedback Manipulation

Now we study how the scheme performs at a low sparsity with feedback manipulations. We consider each of the attack strategies we described earlier and study how different algorithm components perform under each of them.

Similarity Measure. We first study the performance of different similarity measures in terms of trust computation error and coverage under different threat models.

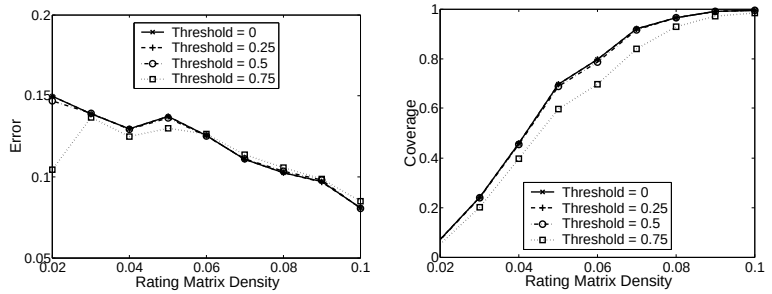
Figure 7 shows the trust computation error and coverage using different similarity measures under non-collusive model where each individual malicious user acts independently. We see that the three similarity measures have the same effect on the computation error which increases as the percentage of malicious users increases. Pearson correlation and Euclidean distance give slightly better precision than vector similarity. This may be contributed to the fact that vector similarity assigns a default 0 value for the missing feedback rating when comparing two users which results in errors. The computation coverage is not affected.

For the collusive threat models, collusive and col-



(a) Trust Computation Error (b) Trust Computation Coverage

Figure 5: Trust Computation Error and Coverage of Different Path Combination Functions with Varying Sparsity Levels



(a) Trust Computation Error (b) Trust Computation Coverage

Figure 6: Trust Computation Error and Coverage of Varying Neighbor Selection Threshold with Varying Sparsity Levels

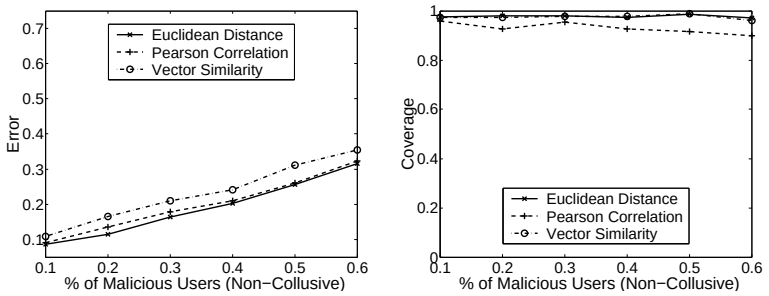
lusive camouflage, we assume the goal of the collusive malicious users is to boost their own ratings. In particular, in the collusive camouflage model, malicious users camouflage as honest users by providing honest ratings about the users outside their group in order to be similar to other users so that their dishonest (high) ratings about malicious users themselves can be counted. So in addition to measure the trust computation errors for the whole population of users, we also measure the computation errors for the malicious users which will reflect how the collusive camouflage users achieve their malicious goal.

Figure 8 and 9 show the trust computation error and coverage using different similarity measures under collusive, and collusive camouflage threat model respectively. We observe that by colluding with each other, the malicious users are able to decrease the system computation accuracy to a large extent esp. when there is a majority of them in the system. The target error is even higher than the general error. Among the different similarity measures, Pearson correlation is slightly more resistant to malicious users than the other two measures and vector similarity suffers the most. Another interesting phenomenon is that by using a collusive camouflage strategy, the malicious users are able to increase the target error to a larger extent than the collusive model, meaning their own ratings may be raised to a larger extent. However, the general error is decreased because the camouflaged malicious users provide honest ratings which actually make good contributions to the community feedback.

We note that the system manages to be effective until about 50% of the users are malicious even in the worse collusive model. This is quite acceptable as a system may be useless anyway when a majority of the community is malicious.

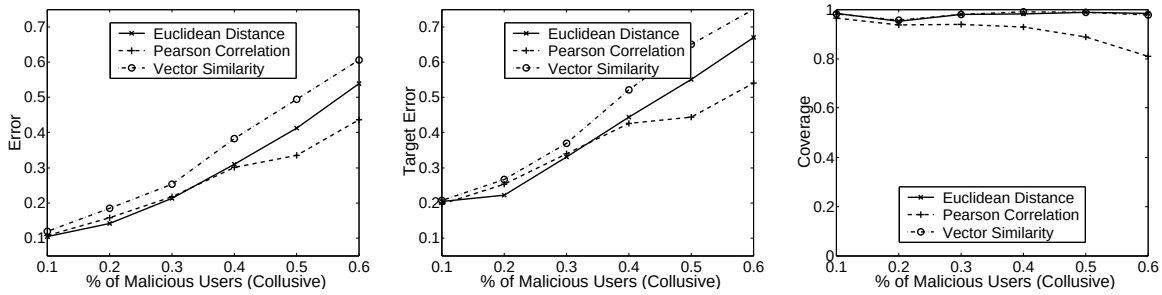
Maximum Path Length. We next study the effect of neighbors' maximum path length on the trust computation error and coverage with a sparse feedback under different threat models.

Figure 10 shows the trust computation error and coverage under non-collusive model. We can again make a few interesting observations. First, the 2-Hop algorithms and above provide significantly better pre-



(a) Error (b) Coverage

Figure 7: Trust Computation Error and Coverage of Different Similarity Measures in Non-Collusive Model



(a) Error (b) Target Error (c) Coverage

Figure 8: Trust Computation Error and Coverage of Different Similarity Measures in Collusive Model

cision and coverage than the basic 1-Hop algorithm. This shows that considering indirect neighbors is important in countering sparse data and as a result it also provides a better resilience to the manipulation of feedback by malicious users. Second, 3-Hop and 4-Hop algorithms do not perform much better than 2-Hop algorithm. This shows that considering neighbors that are far away does not help much in coping with malicious feedback manipulation.

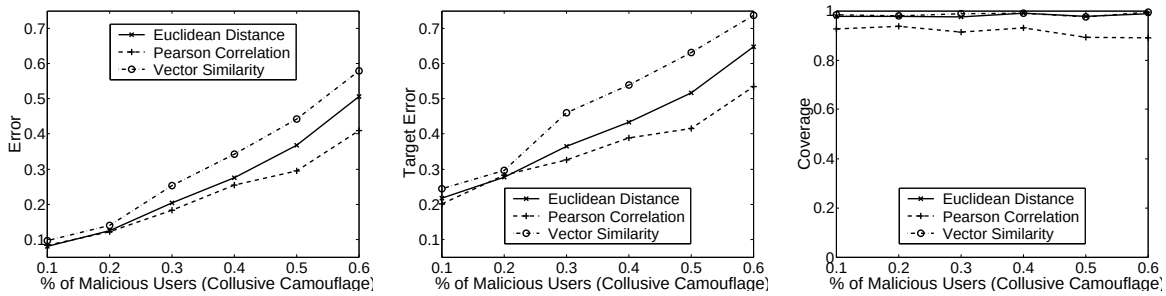
Figure 11 and 12 show the trust computation error and coverage results under collusive and collusive camouflage threat model respectively. We see a similar trend as in the non-collusive model but to a larger extent. Also, the collusive camouflage model is able to slightly increase the trust computation error for the malicious group.

Path Combination Function. We also experimentally studied the effect of different path combination functions in each of the threat models. The results show that the three path combination functions have a similar effect on the trust evaluation error which increases as the percentage of malicious users increases. The minimum function performs slightly worse than the other two similar to what we have observed in their effects on countering sparsity. The collusive models show a similar trend as in the non-collusive model but to a larger extent. Due to space limitations, we omit the graphs for this study in this paper and refer interested readers to [29] for a complete description of the results.

Threshold-Based Neighbor Selection. Finally we studied how threshold-based neighbor selection help countering feedback manipulation attacks. We observe that when we increase the threshold, the precision of the algorithm increases as it helps to filter out malicious peers when aggregating opinions. However, as we have expected, a higher threshold also decreases the coverage of the reputation computation. The graphs are also omitted due to the space restrictions.

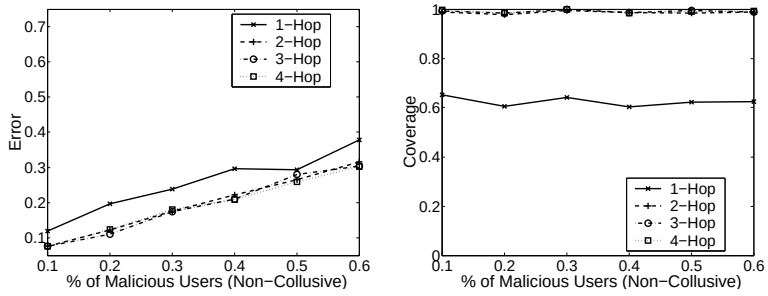
5 Related Work

A number of reputation based trust systems have been proposed recently for P2P networks and web applications [1, 8, 9, 15, 23, 31, 33, 11]. They differ in a variety of aspects, including application domains, inference methodologies, and implementation strategies. There have been a few efforts trying to classify the



(a) Error (b) Target Error (c) Coverage

Figure 9: Trust Computation Error and Coverage of Different Similarity Measures in Collusive Camouflage Model



(a) Error (b) Coverage

Figure 10: Trust Computation Error and Coverage of Varying Maximum Path Length in Non-Collusive Model

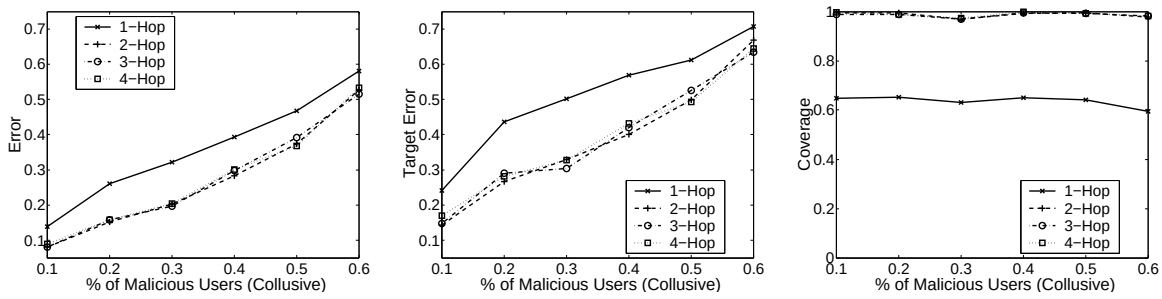
trust schemes based on various dimensions [36, 35, 18]. We first discuss the main techniques related to the feedback vulnerabilities that we study. We then discuss a few representative works that are closely related to our work and examine their approaches and limitations.

A few reputation and trust systems use inference schemes to propagate trust through network by assuming certain transitivity of trust relationships [15, 23, 11]. These schemes typically involve a high computation cost. On the other hand, Marti [17] suggested limited reputation sharing. While it has benefits in load balancing but it is not feasible when the feedback data is sparse.

A common strategy to combat the dishonest feedback is to use certain metric to weigh the information and opinions collected from other peers. A user is much more likely to trust a credible feedback source or trusted acquaintance than from a stranger. There have been a few classes of metrics that have been used. While some works have assumed the recommendation trust is predefined by users [23, 11], one common way is to use the peer's previously determined reputation score to weigh its feedback. The latter do not differentiate recommendation trust (credibility) and service trust (reliability) and are susceptible to front peers attacks [17]. We discuss representative ones including EigenTrust [15] and limited reputation sharing [17] below.

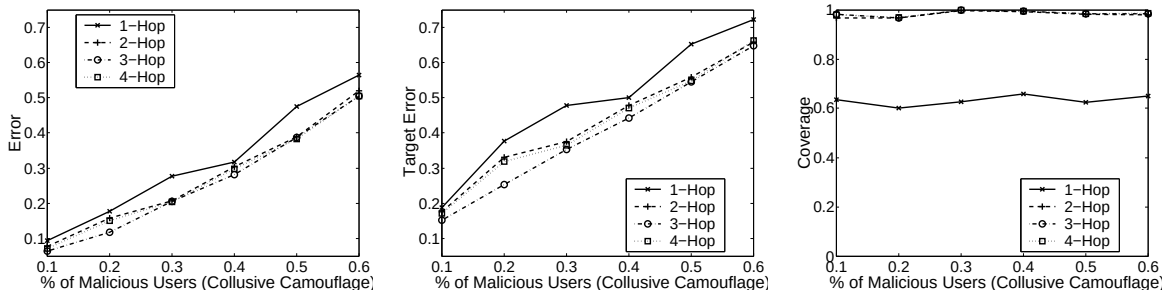
EigenTrust [15] is based on the notion of transitive trust. It propagates the service trust through the whole network until it results in the Eigenvector. They overcome the problem of front peers attacks by assuming there are pre-trusted peers in the network and each peer has to place at least some trust in the pre-trusted peers. This breaks up potential malicious collectives as well as guarantees convergence of the algorithm mathematically. While the algorithm showed promising results, we argue that the pre-trusted peers may not be available in all cases and it remains a research question how to choose pre-trusted peers in a network. Another shortcoming of their approach is that the implementation of the algorithm is very complex due to the global trust propagation scheme and requires strong coordination and synchronization of peers.

Limited reputation sharing scheme proposed by Marti et al. [17] uses only limited or no informa-



(a) Error (b) Target Error (c) Coverage

Figure 11: Trust Computation Error and Coverage of Varying Maximum Path Length in Collusive Model



(a) Error (b) Target Error (c) Coverage

Figure 12: Trust Computation Error and Coverage of Varying Maximum Path Length in Collusive Camouflage Model

tion sharing between nodes. The rating of a peer is the combination of the local statistics and the quorum rating, which takes a weighted average of opinions of other peers. A larger weight is given to the nodes that have a high reputation trust value by behaving well and providing good files. In contrast to global history schemes, the focus is to use limited sharing of opinions. They studied some interesting effects related to load-balancing and message traffic in the P2P network. A problem with this scheme, as the paper shows, is that it can be easily defeated by a group of malicious peers who set up front nodes that properly trade only authentic files, but when asked for their opinion of other nodes, only promote malicious nodes in the group. This problem can be alleviated when peers consider their own local statistics and give very small consideration of other peers' opinions. However, when the input matrix is sparse, majority of the nodes will not have statistics for most of other nodes.

Yu et al. proposed a distributed reputation mechanism for P2P networks with techniques for detecting deception in reputation propagation and aggregation [31, 33]. The weighted majority techniques they adopted is in spirit similar to ours in that it associates a credibility weight with each agent when aggregating their opinions (differentiated from service trust). However, in contrast to our work, all the weights are initialized to 1 in their approach and they only get updated (downgraded) after unsuccessful predictions. In other words, the bad peers only get recognized (having their credibility weights downgraded by other peers) after their dishonest feedback incurs a transaction between a malicious peer and an honest peer. The malicious peers who badmouth good peers may never get recognized as the good peers who get malicious dishonest feedback may not get a chance to interact with other peers. In our approach, the derived similarity automatically reflects the changes in the ratings. It would be interesting to explore the possibilities to allow the credibility to be updated by users as in their work.

Another related area of research is recommender systems based on collaborative filtering techniques. Adomavicius et al. [3] provides a latest survey of the state-of-the-art in recommender systems. While reputation systems and recommender systems are closely related and we adopted certain techniques from collaborative filtering, there are also important differ-

ences between them. First, recommender systems are concerned with items or products that are fairly static while reputation systems deal with transactions and need to cope with dynamic behaviors of malicious peers [18, 26]. S. K. Lam and J. Riedl [16] experimentally studied several types of shilling attacks on collaborative filtering systems. We benefited from their attack designs and modeled and analyzed feedback manipulation attacks on reputation systems. There has also been some research that attempted to alleviate the sparsity problem in collaborative filtering. Dimension reduction techniques aim at reducing the dimensionality of the user-product matrix. Empirical results have shown that dimension reduction techniques improve recommendation performance in some applications but perform poorly in others [24] as potentially useful information might be lost during the reduction process. Huang et al. [14] modeled the user-product interactions as a bipartite graph and modeled the recommendation problem as association retrieval problem However, their framework only deals with systems that have binary transactional data. Most importantly, they did not consider the potential vulnerabilities of the system.

6 Conclusion

We presented in this paper a similarity inference scheme that helps the a reputation system coping with feedback sparsity with potential feedback manipulations. We studied various algorithmic components in the scheme including the similarity measure and the similarity inference model. Our experimental evaluations show that by considering indirect neighbors, the scheme provides resilience for the trust framework against feedback sparsity even with various feedback manipulation attacks.

Our work continues along several directions. First, we would like to conduct a formal study analyzing the effect of various algorithmic parameters, for example, to determine the optimal maximum path length for a given sparsity and distribution of the community. Second, the proposed similarity inference scheme essentially clusters similar users together based on their feedback and helps them to evaluate each other despite sparse feedback. An interesting and opposite phenomenon that has been suggested recently is that when recommendations are generated in a distributed manner with scattering, the quality of the network could improve when clusters are reduced [25, 34]. This corresponds to the intuition that people benefit from knowing others outside their parochial groups. This suggests an interesting research opportunity that considering dissimilarity as well as similarity in the inference scheme may yield unexpected yet potentially good results. Finally, we are also interested in applying the techniques to web spam filtering and service reputation ranking in semantic web.

References

[1] K. Aberer and Z. Despotovic. Managing trust in a peer-to-peer information system. In *ACM CIKM*, 2001.

[2] E. Adar and B. A. Hubeman. Free riding on gnutella. *First Monday*, 5(10), 2000.

[3] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6), 2005.

[4] T. Beth, M. Borcherding, and B. Klein. Valuation of trust in open networks. In *3rd European Symposium on Research in Computer Security (ESORICS)*, 1994.

[5] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *14th Conference on Uncertainly in Artificial Intelligence*, 1998.

[6] H. Chen and D. T. Ng. An algorithmic approach to concept exploration in a large knowledge network (automatic thesaurus consultation): symbolic branch-and-bound search vs. connectionist hopfield net activation. *Journal of American Society of Information Science*, 46(5), 1995.

[7] A. M. Collins and E. F. Loftus. A spreading activation theory of semantic processing. *Psychology Review*, 82(6), 1975.

[8] F. Cornelli, E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. Choosing reputable servents in a p2p network. In *11th International Conference on World Wide Web*, 2002.

[9] E. Damiani, S. Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *CCS*, 2002.

[10] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *ACM Electronic Commerce Conference*, 2004.

[11] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *13th International Conference on World Wide Web*, 2004.

[12] Z. Gyongyi and H. Garcia-Molina. Combating web spam with trustrank. In *30th International conference on Very Large Databases (VLDB)*, 2004.

[13] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR*, 1999.

[14] Z. Huang, H. Chen, and D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions in Information Systems*, 22(1), 2004.

[15] S. Kamvar, M. Scholsser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *12th International Conference on World Wide Web*, 2003.

[16] S. K. Lam and J. Riedl. Shilling recommender systems for fun and profit. In *13th International Conference on World Wide Web*, 2004.

[17] S. Marti and H. Garcia-Molina. Limited reputation sharing in peer-to-peer systems. In *ACM Electronic Commerce Conference*, 2004.

[18] S. Marti and H. Garcia-Molina. Taxonomy of trust: Categorizing p2p reputation systems. *COMNET Special Issue on Trust and Reputation in Peer-to-Peer Systems*, 2005.

[19] E. M. Maximilien and M. P. Singh. Conceptual model of web service reputation. *SIGMOD Record*, 31(4), 2002.

[20] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, 1998.

[21] L. Ramaswamy and L. Liu. Freeriding: A new challenge for peer-to-peer file sharing systems. In *36th Annual Hawaii International Conference on System Sciences (HICSS-36)*, 2003.

[22] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara. Reputation systems. *Communications of the ACM*, 43(12), 2000.

[23] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *2nd International Semantic Web Conference*, 2003.

[24] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Application of dimensionality reduction in recommender systems: A case study. In *WebKDD workshop at ACM SIGKDD*, 2000.

[25] M. P. Singh, B. Yu, and M. Venkatraman. Community-based service location. *Communications of the ACM*, 44(4), 2001.

[26] M. Srivatsa, L. Xiong, and L. Liu. Trustguard: countering vulnerabilities in reputation management for decentralized overlay networks. In *14th International conference on World Wide Web*, 2005.

[27] V. Swarup and J. T. Fabrega. Trust: Benefits, models, and mechanisms. In *Secure Internet Programming: Security Issues for Mobile and Distributed Objects, Lecture Notes in Computer Science*. New York: Springer-Verlag, 1999.

[28] L. Xiong and L. Liu. Peertrust: supporting reputation-based trust in peer-to-peer communities. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16(7), 2004.

[29] L. Xiong, L. Liu, and M. Ahamad. Countering sparsity and vulnerabilities in reputation systems. Technical report, Georgia Institute of Technology, College of Computing, 2005.

[30] R. Yahalom, B. Klein, and T. Beth. Trust relationships in secure systems–A distributed authentication perspective. In *IEEE Computer Society Symposium on Research in Security and Privacy*, 1993.

[31] B. Yu and M. P. Singh. Detecting deception in reputation management. In *2nd International Joint Conference on Autonomous Agents and Multi-agent Systems*, 2003.

[32] B. Yu and M. P. Singh. Incentive mechanisms for peer-to-peer systems. In *2nd International Workshop on Agents and Peer-to-Peer Computing*, 2003.

[33] B. Yu, M. P. Singh, and K. Sycara. Developing trust in large peer-to-peer systems. In *First IEEE Symposium on Multi-Agent Security and Survivability*, 2004.

[34] B. Yu, M. Venkatraman, and M. P. Singh. An adaptive social network for information access: Architecture and experimental results. *Applied Artificial Intelligence*, 17(1), 2003.

[35] Q. Zhang, T. Yu, and K. Irwin. A classification scheme for trust functions in reputation-based trust management. In *ISWC Workshop on Trust, Security, and Reputation on the Semantic Web*, 2004.

[36] C.-N. Ziegler, G. Lausen, and U. Freiburg. Spreading activation models for trust propagation. In *IEEE International Conference on e-Technology, e-Commerce and e-Service*, 2004.