Countering Feedback Sparsity and Manipulation 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*—Reputation systems provide a promising way for building trust through social control in collaborative communities by harnessing the community knowledge 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 two closely related problems - feedback sparsity and potential feedback manipulations - and propose a feedback similarity based inference framework. We perform extensive evaluations of various algorithmic components of the framework and evaluate their effectiveness on countering feedback sparsity in the presence of feedback manipulations.**

I. INTRODUCTION

Reputation systems [19] provide attractive techniques for building trust in a variety of collaborative communities and web applications. By harnessing the community knowledge in the form of feedback, they help participants decide who (what) to trust, encourage trustworthy behavior, and deter dishonest participation [27]. Many electronic markets and online communities have reputation systems built in, such as eBay[1], resellerratings[2], Yahoo! Auction[3], and Slashdot (Karma[4]). Most of them aggregate users' feedback into a reputation score through a summation or average over a period of time. Reputation systems with more sophisticated aggregation methods have been proposed recently for peer-to-peer communities [1], [10], [22], [32], [24], [34], [6], [25], [12] and the (semantic) web [28], [15].

Feedback Sparsity and Feedback Manipulation. Reputation systems rely on historical transaction and feedback data to derive a reputation score for entities. In large scale peer-to-peer communities, each peer may have interacted with only a small number (percentage) of peers and hence has insufficient feedback about and from other peers. When the feedback is sparse, the system can not evaluate the trust value of most peers or the evaluations suffer from lack of accuracy. In addition, a piece of feedback is simply a statement from a user about another user, typically, the service (or information) consumer about the service (or information) provider. There is no mechanism to guarantee that the statement is honest. Malicious users may manipulate the feedback data in order to benefit themselves or damage the system.

Surprisingly, while there is work dedicated to each of the problems, very few have studied the feedback sparsity problem with potential manipulations of feedback. Our primary viewpoint in this paper is that the two problems are closely related. Feedback sparsity concerns with the quantity of feedback. Feedback manipulation concerns with the quality of feedback and affects the quantity of good feedback. As a result, the two problems may have a magnifying effect on each other. When adversaries pollute the feedback data, the valid or usable feedback becomes sparse, which makes sparsity problem worse. On the other hand, when feedback is sparse, feedback manipulation attacks may be magnified and have a more detrimental effect.

Current Techniques and Research Challenges. Some studies have been dedicated to dishonest feedback along with other vulnerabilities in reputation systems such as fake transactions and dynamic behaviors [22], [32], [30]. A main technique that has been proposed to cope with dishonest feedback is to associate a credibility weight with each peer (as versus the service reliability of the peer). A peer who has consistently provided honest feedback should be associated with a higher credibility weight so the system can differentiate honest feedback from dishonest ones. Collaborative filtering systems [3] suggested user similarity (based on previous ratings about common items) as an effective measure in identifying users with similar interests and predicting a user's rating about a certain item. User similarity (based on previous feedback about common peers) has been also used as a personalized credibility measure in reputation systems against certain attacks including collusion among a group of peers who provide good ratings within the group and bad ratings outside the group [32]. Unfortunately this does not solve but rather exacerbates the spartiy problem: in addition to the limited personal feedback, common set of peers that two users have both interacted with is likely small which will result in poor similarity quality.

Research in trust inference [33], [5], [31], [14], [20], [13] addresses trust propagation of initial trust relationships assuming certain transitivity of trust. They differ with reputation systems in that they assume the initial trust relationships are predefined among nodes and use graph theoretic models or

[1]http://pages.ebay.com/help/feedback/
[2]http://www.resellerratings.com/
[3]http://help.yahoo.com/l/us/yahoo/auctions/general/agen-07.html
[4]http://slashdot.org/faq/com-mod.shtml#cm700

matrix operations to propagate the initial trust. Some recent works adopted inference techniques in reputation systems. A notable work EigenTrust [22] derives initial trust based on personal feedback and performs a global trust propagation till it finds the eigenvector of the initial trust matrix. However, by treating the service trust the same as credibility for propagation, the model is vulnerable to front peer attacks where adversaries exploit the trust transitivity property by creating false trust links [24]. There also has been research on collaborative filtering systems [3] that addresses the sparsity or cold start problem by graph theoretic approaches [4], [29], [18]. However, they did not consider the potential feedback manipulations.

While these works shed light on the potential adoption of inference techniques to address sparsity issues in reputation systems, a few research challenges remain. First, what initial relationships among nodes do we use to perform inference? 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, how does the inference model cope with potential feedback manipulations?

Contributions and Organizations. Bearing these questions in mind, we propose a similarity inference framework and study experimentally how different inference parameters perform in countering feedback sparsity and feedback manipulation. The paper makes a number of unique contributions. First, we show there are important relationships between feedback sparsity and feedback manipulation. We explore different attack strategies related to feedback manipulation and quantify the sparsity problem (Section II). Second, we extend our earlier work on using similarity as credibility measures [32] and apply inference on the credibility to cope with the sparsity problem in the potential presence of dishonest feedback (Section III). While the inference scheme builds on general graph-based inference techniques, there are two key ideas that extend existing work: 1) the initial trust relationship is derived from the similarity based credibility as opposed to personal feedback. We argue that propagating similarity based trust (credibility) avoids the attacks introduced by propagating personal feedback based trust (service reliability), and 2) the propagation is performed in a local manner. Finally, we study experimentally how variant algorithmic components of the framework such as inference function and depth of inference perform in coping with sparsity in the presence of feedback manipulation (Section IV). We conclude the paper by a review of the related work, a summary and a brief discussion of future research (Section VI).

II. 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 and feedback manipulation.

A. Terms and Definitions

A *peer-to-peer community* consists of N peers who perform transactions with each other such as eBay. It can be also generalized into a community that consists of N users and M service (or information) providers such as in resellerratings.com. Each *transaction* has a client (service consumer) and a server (service provider). A peer may serve as a client in one transaction and as a server in another. A *transaction feedback* is a statement issued by the client about the quality of service (or information) provided by the server in a single transaction. A *personal feedback* is a user's general impression about a service provider based on its personal experiences with the server. It can be derived from all previous transaction with the server such as the percentage of positive transactions. We can represent all the personal feedback in the network as a user-user feedback matrix, where each cell represents a personal feedback from a given user about a given server. The goal of a reputation system is to compute a *reputation trust* or *trustworthiness* value for a given service provider. We refer to the evaluating user as *source* and the user to be evaluated as *target*. Without loss of generality, we define the reputation trust as a combination of the user's personal feedback and an aggregation of the community feedback. *Credibility* of a user a indicates how credible a is in providing feedback. In contrast, trustworthiness indicates how reliable a is in providing service. Some works use trustworthiness as a general notion. We argue that feedback trustworthiness (credibility) should be differentiated from service trustworthiness (reliability) as user b may trust user a for its ratings but not necessarily its service and vice versa.

B. Threat Model

In a common feedback manipulation attack, malicious users provide false feedback to boost their own ratings or decrease the ratings of other peers so they will be chosen by other peers even when their services are unsatisfactory. We consider various dimensions of this type of attacks in this section[5].

Attack Goals. We consider two different attack goals depending on the target.

- *Random attacks*: malicious users try to reduce the overall performance of a system as a whole and do not target any particular users.
- *Target attacks*: malicious users try to force the ratings of a target user to a particular target reputation value. For example, in a push (nuke) attack, the goal is to force all predicted ratings of targeted users to the maximum (minimum) rating.

Attack Models. We also consider two attack models depending on whether malicious users collude with each other.

- *Non-collusive model*: individual malicious users do not know each other and each do something bad, hoping it will affect the system.

[5]Other attack incentives such as free riding [2], [21] and dynamic behaviors [30] are not considered in this paper.

- *Collusive model*: multiple malicious peers may form a group and collude with each other in order to achieve their goal that is typically targeted towards boosting the ratings of the whole or part of the group.

A related attack where an individual user creates multiple fake profiles that act as a collusive group can be modeled similarly. We do assume adversaries have to pay a cost to create a profile so it is not feasible to create large number of 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 collusive attacks may be specifically designed to exploit a particular weakness in a specific algorithm or class of algorithms. For example, for systems that do not differentiate service trust (reliability) and feedback trust (credibility), one simple strategy is to have part of the peers act as front peers or moles [24]. These peers always cooperate with other peers in order to increase their reputation and then provide misinformation to promote other malicious peers.

Differentiating reliability and credibility helps reputation systems to avoid front peer attacks and are more robust to dishonest feedbacks. Our previous work PeerTrust [32] studied dishonest feedback attacks in a non-collusive and a naive collusive model with dense feedback data. In this paper, we will study more sophisticated attack strategies targeted specifically at similarity based models inspired by the shill attack designs in recommender systems [23].

- *Collusive*: a straightforward strategy is for collusive users to rate each other with the maximum rating and rate other peers with a minimum rating. The hope is that as they boost their own ratings, they also decrease other peers' ratings or damage the performance of the reputation system. However, they may end up with low similarity to the normal users and in turn their ratings will not be counted as much.
- *Collusive Random*: another way is for collusive users to rate each other with the maximum rating and rate other peers randomly. The hope is that as they boost their own ratings, they are also somewhat similar to other users and they also damage the performance of the reputation system by providing random ratings.
- *Collusive Camouflage*: a more sophisticated strategy is for collusive users to rate each other with maximum rating but rate other peers honestly so that they will be similar to more honest users, and thus, have a larger effect on boosting their own ratings. Hypothetically, this strategy will allow adversaries to mount a more detrimental attack and boost their ratings by camouflaging as honest users.

C. Sparsity Problem

Now we consider the sparsity problem for using a similarity based trust scheme [32]. A source peer a computes the reputation of a target peer j as a weighted average of j's ratings from other peers, where the weight is a personalized similarity measure between source peer a and each other peer b who gives a rating about j. We model the feedback by a and b over a common set of peers for which both a and b have rated as two vectors and compute the similarity between the two vectors. When feedback is sparse, two given peers may have a very limited number of or zero co-rated peers and the similarity can not be derived. The reputation can not be computed if the similarity can not be derived for all the users who have rated the target peer.

Formally, the probability of an undefined trust computation is the probability that all of the users who have rated target peer j do not have co-rated peers with a such that the similarity cannot be computed. Let N denote the number of peers in the community, and r be the average number of ratings each user receives, the reputation computation coverage is given by Equation 1 where $\frac{\binom{N-r}{r}}{\binom{N}{r}}$ models the probability of two peers not having co-rated peers.

$$Coverage(N, r) = 1 - \left(\frac{\binom{N-r}{r}}{\binom{N}{r}}\right)^r \qquad (1)$$

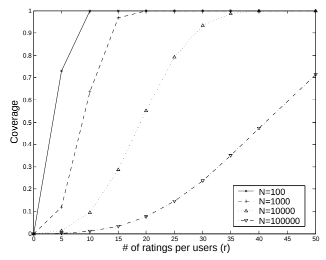


Fig. 1: Sparsity Problem

While N, the number of users, could be of the order of thousands, r, the number of ratings per user, could be very small, and the resulting coverage can be very low. Figure 1 plots Equation 1 with varying total number of users and average number of ratings per user. When a new user joins the community, it does not have many ratings about or from other users. This is also referred to as cold start problem in recommender systems [4], [29].

III. 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 with presence of dishonest feedback.

A. Overview

We first model our problem in a graph theoretic way. Suppose each peer is represented as a node in the graph. A personal feedback by peer d about peer e is represented by a directed link from d to e. 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. If we treat all the ratings each peer has given to others as a vector, we can derive similarity between users in terms of their feedback filing and use this as a personalized credibility measure. Figure 2 illustrates a partial personal rating matrix and a corresponding graph with solid links representing personal feedback and dashed links representing similarity between users.

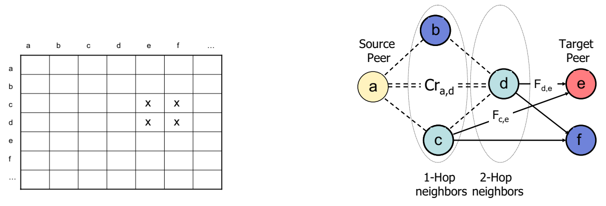


Fig. 2: Similarity Inference: Illustration

When the feedback data is sparse, it is not always possible to derive the direct similarity between users (recall Section II-C). This motivates us to explore the transitive associations of similarity. If a user a is similar to b, b is similar to d, a should be somewhat similar to d. The indirect similarity link is represented as double dashed link in Figure 2.

The task of a reputation system is to evaluate the trustworthiness of a target peer e by aggregating its incoming links for a source peer a. The framework uses feedback similarity as a personalized credibility weight in feedback aggregation and uses a similarity inference scheme to compute the weight for indirect neighbors. The main steps are: 1) compute initial trust (personal ratings matrix) based on personal experience, 2) compute direct similarity based on the ratings matrix, 3) find indirect neighbors and compute indirect credibility, and 4) aggregate user feedback (possibly from a selected neighborhood of users) based on the credibility of the users. While the inference scheme builds on general graph-based inference techniques [28] and is not completely novel, there are two key ideas: 1) the initial trust relationship is derived from the similarity based credibility as opposed to personal feedback, and 2) the propagation is performed in a local manner. We explain each of the steps in subsequent subsections and discuss the implications of various algorithmic components in the setting we consider.

B. Computing Credibility using Similarity

We first derive the weight of direct links between two users a and b. Assuming the personal feedback $F_{a,i}$ 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 [30] or to include transaction contexts such as various transaction types and sizes [32]. If we treat the ratings each peer has provided in the past about other peers as a vector, we can compute similarity between the rating vectors of two users, denoted as I_a and I_b. It is worth mentioning that the rating vectors used to compute similarity need to be in the same service context. 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 the most widely used and computes the degree of linear relationship between two variables.

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

- *Vector cosine similarity*. The vector similarity is another widely used measure and is calculated as the cosine of the angle between the two corresponding vectors (normalized inner product):

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

- *Euclidean distance*. The Euclidean distance between the two feedback vectors can be also used for deriving the similarity measure [32].

$$Cr_{a,b} = \sqrt{\sum_{i \in I_a \cap I_b} (F_{a,i} - F_{b,i})^2} \quad (4)$$

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 [7], [16] 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. However, Pearson correlation has a potential drawback as it assumes an approximate Gaussian distribution of the data and may not be robust with respect to outliers. We will study experimentally how these measures perform in our context with sparse feedback and potential feedback manipulations in Section IV.

C. Similarity Inference

We refer to all users who have a direct similarity weight defined with a as a's direct similarity neighbors. When a needs to evaluate the trustworthiness of another peer e, it asks its neighbors for their opinions about e and aggregates their opinions. (We will discuss neighborhood selection techniques later in this section). When the input data is sparse, each peer may have 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 (recall Equation 1). In this section, we present algorithms to find indirect neighbors of a given user and propagate the similarity weight between the user and indirect neighbors.

Discovering Indirect Neighbors. We can model the problem as a transitive closure problem to find all nodes that are reachable by node a 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 a. If a node is reachable by a by at least l hops, we call the node a's l-hop neighbor. a's 1-hop neighbors are also a's direct neighbors. Figure 2 shows 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.

Computing Indirect Weights. For a's indirect neighbors, we need to compute an indirect weight. For a l-hop neighbor b, we consider all l-length paths from a to b and compute an indirect weight based on these paths.

Formally, for each path, we infer an indirect weight for this path as a product of the direct weights on individual links. 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 to 1 in our implementations.

There are some important implications in propagating similarity as a trust relationship. Using production as the path concatenation function assumes a pessimistic behavior. One particular situation is when node a distrusts b and b distrusts c because of a low similarity, a will choose to further distrusts c. Alternatively, distrust could be propagated separately from trust with different functions [14].

If there are multiple paths of length l from node a to node b, we need a function to select or combine the inferred values from the paths into a single value. We consider three such functions: maximum, minimum, and average [28].

- *Maximum*. 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 believes j. This will also potentially help balance the pessimistic property of the production propagation on a single path.
- *Minimum*. 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 a will only believe another user b to an extent with which all its closer neighbors believe b.
- *Average*. 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 is harder to implement using standard algorithms and we will discuss the implementation later in this section.

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 has been observed in collaborative filtering systems that the neighbors with low correlations will greatly affect the prediction accuracy [7]. So we can use neighborhood selection techniques suggested in collaborative filtering systems to select a subset of neighbors, namely, threshold based selection that selects neighbors with absolute similarity greater than a certain threshold, and k nearest neighbor selection.

When such schemes are used, the threshold 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 similarity graph by an adjacency matrix M with each cell representing the similarity weight, assuming the weight is normalized for each user, we can also use matrix production M^l to compute the indirect weight within l hops. It essentially corresponds to using production as the path concatenation function and using weighted average as the path combination function. We can also interpret the similarity inference problem as a stochastic transition process similar to the random walker model in PageRank [26] and EigenTrust [22]. Imagine the source user a is trying to find similar neighbors, at each step, it hops to a neighbor according to the given probability (normalized similarity with the neighbor). The difference between our approach and web of trust or EigenTrust is that user a stops when it reaches the maximum path length while in the latter approach a user's personal beliefs can be diluted by the infinite aggregation of other users' beliefs and result in a global trust vector.

It is worth mentioning, however, if we normalize the similarity weight by each user, the similarity value does not have the semantic meaning any more but only a relative rank. So the threshold based neighbor selection will not work with normalized similarity but k nearest neighbor selection will.

Implementation and Practical Implications. 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. It can be conveniently plugged into existing reputation system architectures such as that in [32]. A node a first initializes its neighbor list N_a by adding itself into the list. Then at iteration step l, it gets all the l-hop neighbors and computes their indirect weights based on the given path combination function f. The algorithm terminates after it retrieves all the L-hop neighbors. We can also adopt spreading activation algorithms [9], [8] for more efficient implementations, for example, the branch and bound algorithm, if the path combination functions are associative (out of the functions we considered, maximum and minimum are associative but average is not).

Since the algorithm runs for a bounded number of hops (we show later L=2 is sufficient for many settings), the added cost of computation and communication is minimal considering the sparsity of the network. This is very important for a system that has to deal with dynamic environments including continuously changing user feedback, similarities, and neighbors. Other techniques such as the fading memory scheme developed in our earlier work [30] can be also deployed to summarize and maintain the transaction feedback in an incremental manner and further optimize the implementation.

Algorithm 1 Local Algorithm for Discovering Indirect Neighbors and Computing Indirect Weights Executed at Node a

```
Input: $L$, $f$
Output: $Neighbors_a$ - a list of (node, similarity) pairs
$Neighbors_a$.add(($a$,1))
$l \leftarrow 1$
while $l \leq L$ do
    $NewNeighbors \leftarrow \phi$;
    for all $b \in Neighbors_a$ do
        for all $c \in$ direct neighbors of $b$ do
            $Cr_{a,c} \leftarrow Cr_{a,b} * Cr_{b,c}$
            if $c \notin NewNeighbors$ then
                $NewNeighbors$.add ($c$, $Cr_{a,c}$)
            else
                $NewNeighbors$.update ($c$, $f(N$.get($c$), $Cr_{a,c}$))
            end if
        end for
    end for
    $Neighbors_a$.add($NewNeighbors$)
    $l \leftarrow l + 1$
end while
```

IV. Experimental Evaluations

We performed an extensive set of simulations to empirically study the performance of the framework under different sparsity levels and threat models with various algorithm parameters. We show its benefit compared to PeerTrust [32] in countering sparsity and feedback manipulations. We did not include comparisons with other reputation systems because the advantage of PeerTrust in terms of resisting front peer attacks over reputations systems that do not differentiante service trust and feedback trust has been demonstrated earlier [32], [24]. As a common practice for peer-to-peer reputation systems, we use simulations on synthetic data to cover a broad range of attack scenarios and sparsity levels. It would be interesting to perform experiments on real world peer-to-peer rating datasets such as Epinions web-of-trust data[6] if it becomes publically available.

[6]http://www.epinions.com/help/faq/?show=faq_wot

A. Experimental Design

In the first set of experiments, we study how various algorithmic parameters of the system handle sparse feedback when there is no feedback manipulation. In the second set of experiments, we study how the scheme performs at a fixed low sparsity against feedback manipulation using various attack strategies as described in Section II-B. Table I summarizes the main experimental parameters with their default values used for the experiments. Table II summarizes the behaviors of malicious peers in different attack models. All results reported are averaged over 10 runs.

TABLE I: Experiment Parameters

Parameter	Description	Default
N	# of peers	100
s	Feedback density	5%
k	% of malicious peers	0
	Similarity Measure	Euclidean
L	Maximum Path Length	2
	Path combination function	Average
Cr_{min}	Neighbor selection threshold	0

TABLE II: Attack Models

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

Evaluation Metrics. We evaluate accuracy and coverage of the reputation evaluation. *Accuracy* measures how the system's computed reputation values for a peer differ from the peer's assigned reliability value. We report Root Mean Square Error for the accuracy. For collusive threat models, the goal of the collusive malicious users is to boost their own ratings. So in addition to the error for the whole population of users, we also measure the error for malicious users which will reflect how they achieve their malicious goal. We refer to this error as *target error*. *Coverage* measures the percentage of a test set of peers that the reputation system is able to compute reputation value for.

B. Countering Sparsity

We first study how various algorithmic components in our scheme handle feedback sparsity when there is no feedback manipulation. We vary the personal feedback matrix density and measure the error and coverage of our inference scheme. The density is defined as the percentage of ratings over the total number of matrix entries and is varied between 2% to 10% for our population size. The selection of these parameters is based on observations in recommender systems and e-commerce data [17] as well as our analysis in Figure 1.

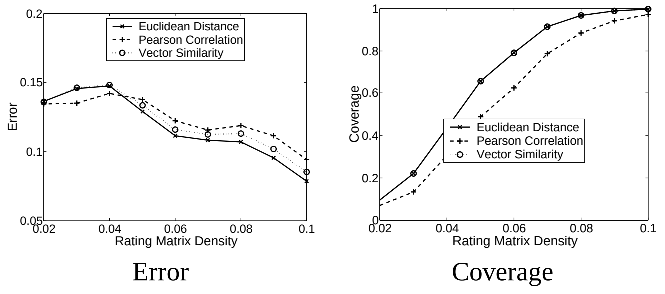


Fig. 3: Effect of Similarity Measure

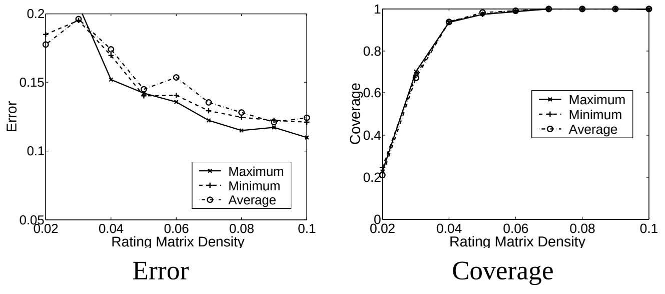


Fig. 5: Effect of Path Combination Functions

Similarity measures. Figure 3 shows the trust computation error and coverage with different similarity measures with respect to varying sparsity levels. The three similarity measures give comparable accuracy and coverage. A noticeable difference is that Pearson correlation has a marginally higher error rate than the other two, due to its assumption of the underlying data distribution. It also suffers from a slightly lower coverage because it can only be computed when two users have commonly rated peers.

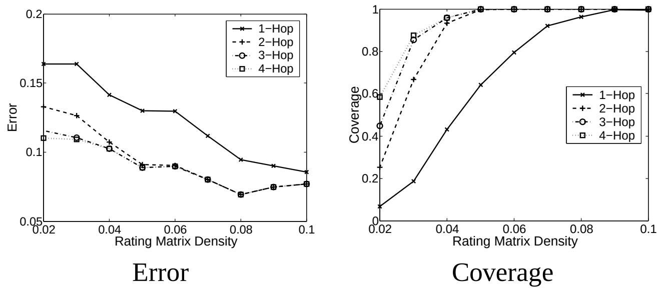


Fig. 4: Effect of Maximum Path Length

Maximum Path Length. Figure 4 shows the trust computation error and coverage with different maximum path lengths under different sparsity levels. Note that 1-Hop algorithm (maximum path length = 1) represents PeerTrust scheme that does not perform trust inference [32]. We can make a few interesting observations. First, the 2-Hop algorithm provides significantly better accuracy and coverage than the 1-Hop algorithm. The performance gain is most significant when the input data is extremely sparse. It shows that considering indirect neighbors is important in countering sparse data. It reflects the intuition that one needs to talk to people outside her close circle to get the best information. Second, 3-Hop and 4-Hop algorithms do not show a significant improvement over 2-Hop algorithm. This can be explained by the small world effect wherein most peers may be connected by a small number of hops (2 in our scenario). It reiterates the observation that limited reputation sharing or a local reputation scheme is desired to avoid the unnecessary computation costs [24]. 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. 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.

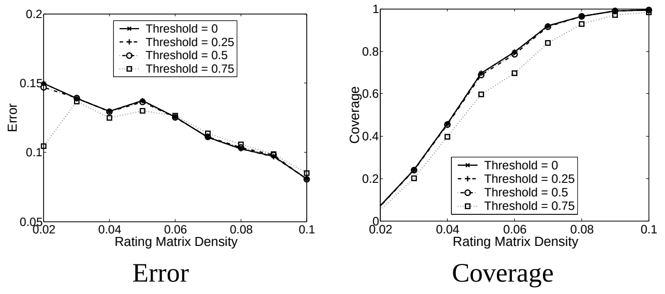


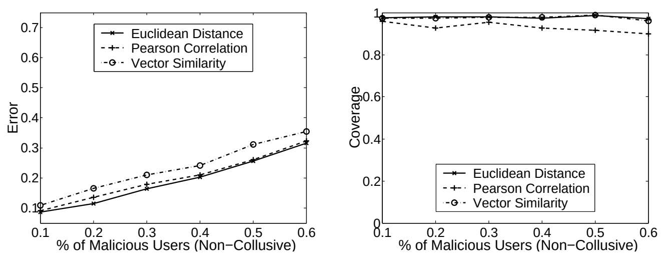
Fig. 6: Effect of Neighbor Selection Threshold

Neighbor Selection Threshold. Figure 6 illustrates the trust computation error and coverage using different threshold values with varying sparsity levels. Interestingly, setting different threshold values does not have a significant impact on the trust accuracy in this case (no feedback manipulation). Having a threshold value of 0.75 increases the accuracy marginally but suffers a lower coverage.

C. Effect of Feedback Manipulation

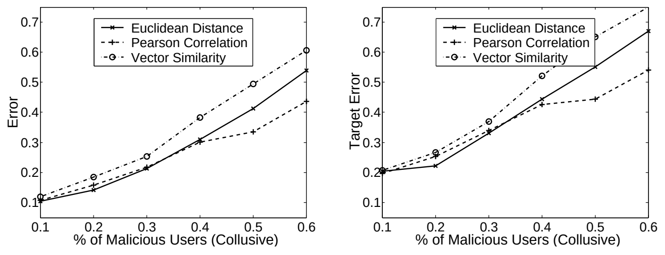
Now we study how the scheme copes with feedback sparsity in the presence of feedback manipulations. We consider each of the attack strategies we described in Section IV-A. The computation coverage graphs for collusive models are similar to those of non-collusive model and thus are omitted due to space restrictions.

Similarity Measure. Figure 7 shows the trust computation error using different similarity measures under different attack models. In the non-collusive model, the three similarity measures have the same trend in computation error which increases as the percentage of malicious users increases. Pearson correlation and Euclidean distance give slightly better



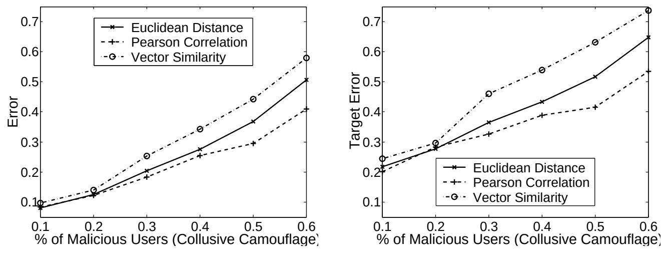
Error Coverage

(a) Non-Collusive Model



Error Target Error

(b) Collusive Model



Error Target Error

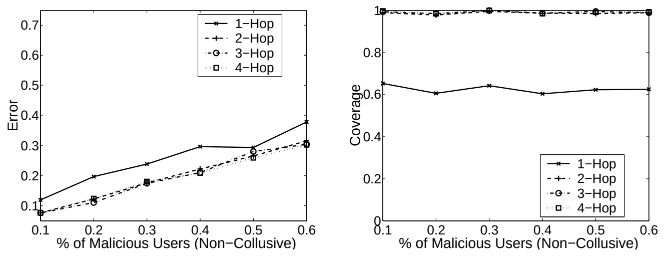
(c) Collusive Camouflage Model

Fig. 7: Effect of Similarity Measures

accuracy than vector similarity. This can be contributed to the fact that vector similarity assigns a default 0 value (potential error) for missing feedback when comparing two users as well as the fact that it normalizes the ratings.

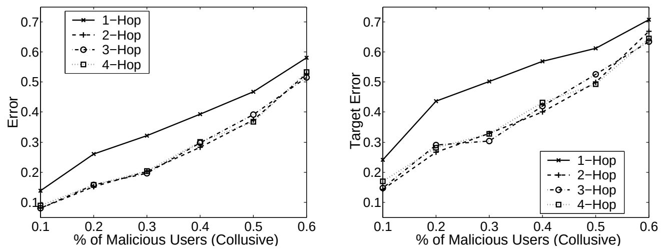
In the collusive models, we observe that by colluding with each other, the malicious users are able to increase the trust evaluation error to a larger extent. This is even more so with target error, indicating that malicious users are able to increase their own ratings to the target rating. Among the different similarity measures, Pearson correlation is slightly more resistant to malicious users 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 slightly compared to the collusive model even though marginal. But on the other hand, the general error is decreased because the camouflaged malicious users provide honest ratings which actually make good contributions to the community feedback.

While the trend of increasing error with increasing percentage of malicious users may not be surprising nor interesting, our goal is to increase the tolerance level of feedback noise or in other words delay the trend. And it is worth noting that the system manages to be effective until over 50% (majority) of the users are malicious even in the worst collusive model.



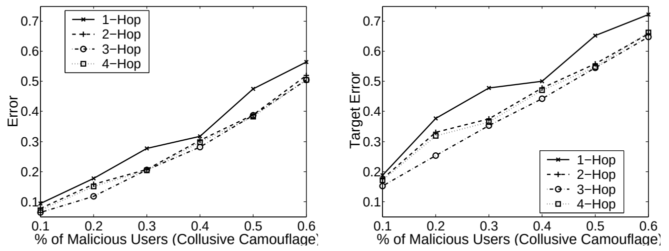
Error Coverage

(a) Non-Collusive Model



Error Target Error

(b) Collusive Model



Error Target Error

(c) Collusive Camouflage Model

Fig. 8: Effect of Maximum Path Length

Maximum Path Length. Figure 8 shows the trust computation error and coverage using various maximum path length under different attack models. The key observations are as follows. First, the 2-Hop algorithms and above provide significantly better accuracy and coverage than the 1-Hop algorithm (PeerTrust 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, we see a similar trend for collusive and collusive camouflage model as in the non-collusive model but to a larger extent. The collusive camouflage model slightly increases the trust computation error for the malicious group.

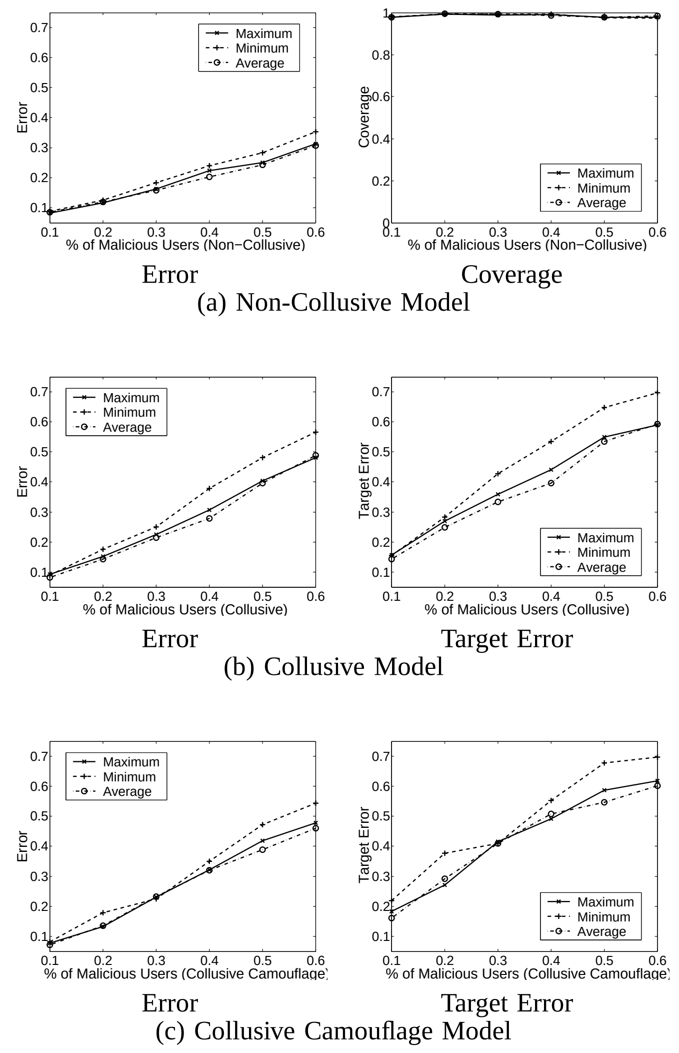


Fig. 9: Effect of Path Combination Functions

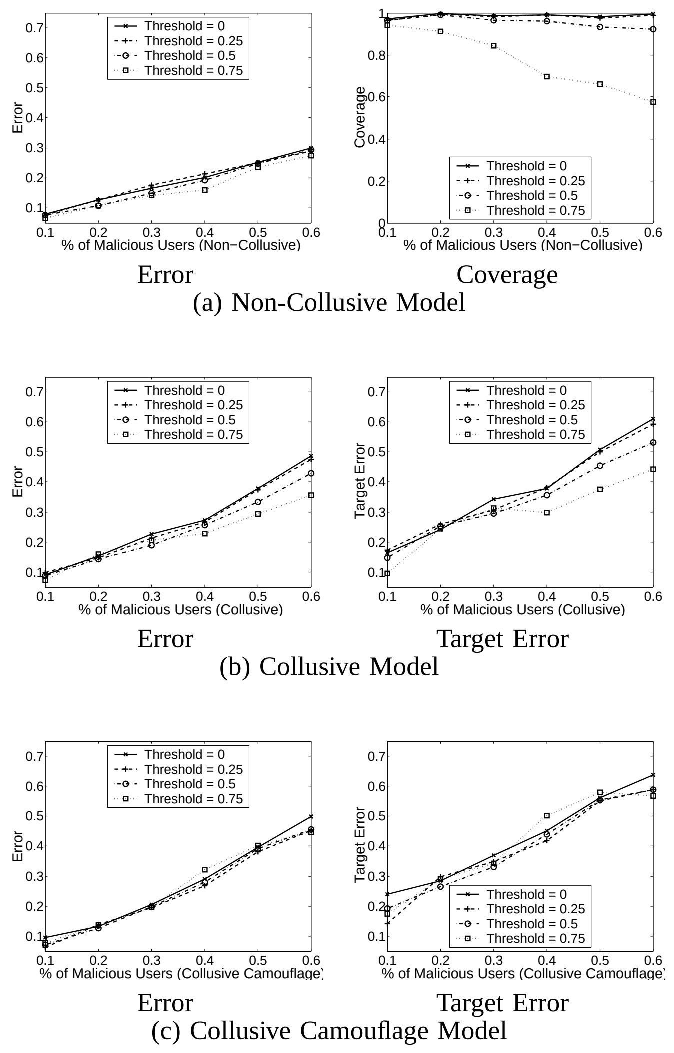


Fig. 10: Effect of Neighbor Selection Threshold

Path Combination Function. Figure 9 shows 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 the case without feedback manipulation. The collusive models show a similar trend as in the non-collusive model but to a larger extent.

Neighbor Selection Threshold. Finally, Figure 10 shows the trust computation error and coverage using different neighbor selection threshold. In contrast to the case without feedback manipulation (Figure 6), we observed that when increasing the threshold, the accuracy of the algorithm increases with a significant extent. This means that a higher threshold helps to filter out malicious peers when aggregating opinions. However, as we expected, it also decreases the coverage of the reputation computation (coverage graphs for the collusive models are omitted).

V. Related Work

A number of reputation systems have been proposed recently for collaborative applications [1], [10], [11], [22], [32], [24], [34], [6], [25], [12], [19]. A few of them use inference schemes to propagate trust through network by assuming certain transitivity of trust relationships [22], [28], [14]. They either assume the recommendation trust is predefined by users [28], [14], or use service trust [22] and are susceptible to front peers attacks. A few proposals advocating the differentiation of credibility from service trust [10], [34], however, the credibility weights of malicious peers only get updated (downgraded) after their dishonest feedback incurs a transaction between a malicious peer and an honest peer. Plus they did not consider sparsity issues.

Research on collaborative filtering systems [3] provides important techniques that apply to reputation systems. It is not surprising that similarity measures can be used as a defensive measure against dishonest feedbacks. In addition, Lam et al.

[23] 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 [4], [29], [18] that attempted to alleviate the sparsity problem in collaborative filtering but most of them did not consider the potential vulnerabilities of the system.

VI. Conclusion

We presented a similarity based inference scheme for countering feedback sparsity with potential feedback manipulations in reputation systems. The key conclusion is that by utilizing the similarity measure as a baseline and a similarity inference of a small number of hops, the framework showed promising resilience against feedback sparsity even with feedback manipulation. Other interesting findings are that Euclidean and Pearson correlation win over Vector similarity which is in fact consistent with findings in general collaborative filtering context [16]. The maximum path combination functions provide more resilience because of its optimistic behavior in combining the beliefs. A higher threshold in neighborhood selection provides a stronger resilience but at the cost of lower coverage.

Our work continues along several directions. We are conducting a formal study to analyze the effect of different attacks and different algorithmic parameters. In addition, we are working with PlanetMath.org[7], an online collaborative encyclopedia on Mathematics, to deploy the proposed system for rating the authors and entries.

Acknowledgement

This research is partially supported by an Emory URC Grant from the first author, and grants from NSF CSR, NSF CyberTrust, NSF ITR, AFOSR, IBM Faculty Award, and an IBM SUR grant from the second author. Our thanks are also due to the reviewers for their comments and suggestions that helped improving the paper.

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] C. C. Aggarwal, J. L. Wolf, K. Wu, and P. S. Yu. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *ACM KDD Conference*, 1999.

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

[6] R. Boutaba and A. Marshall. Management in peer-to-peer systems: Trust, reputation and security. *Computer Networks*, 50(4), 2006.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] Z. Despotovic and K. Aberer. P2p reputation management: Probabilistic estimation vs. social networks. *Computer Networks*, 50(4):485–500, 2006.

[13] J. A. Golbeck. *Computing and applying trust in web-based social networks*. PhD thesis, College Park, MD, USA, 2005. Chair-James Hendler.

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

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

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

[17] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1), 2004.

[18] 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.

[19] A. Josang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, 43(2):618–644, 2007.

[20] A. Josang and S. Pope. Semantic constraints for trust transitivity. In *APCCM '05: Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling*, pages 59–68, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[21] S. Jun and M. Ahamad. Feedex: Collaborative exchange of news feeds. In *15th International conference on World Wide Web*, 2006.

[22] 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.

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

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

[25] S. Marti and H. Garcia-Molina. Taxonomy of trust: Categorizing p2p reputation systems. *Computer Networks*, 50(4), 2006.

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

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

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

[29] 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.

[30] 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.

[31] 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*. 1999.

[32] 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.

[33] 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.

[34] 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.

[7] http://www.planetmath.org