Trustworthiness in networks: A simulation approach for approximating local trust and distrust values

Khrystyna Nordheimer[1], Thimo Schulze[1,2] and Daniel Veit[2]

[1] Chair in Information Systems III,
University of Mannheim,
68131 Mannheim, Germany

[2] Dieter Schwarz Chair of Business Administration,
E-Business and E-Government,
University of Mannheim,
68131 Mannheim, Germany
{nordheimer,schulze}@wifo.uni-mannheim.de

Abstract. Trust is essential for most social and business networks in the web, and determining local trust values between two unfamiliar users is an important issue. However, many existing approaches to calculating these values have limitations in various constellations or network characteristics. We therefore propose an approach that interprets trust as probability and is able to estimate local trust values on large networks using a Monte Carlo simulation method. The estimation is based on existing indirect trust statements between two unfamiliar users. This approach is then extended to the SimTrust algorithm that incorporates both trust and distrust values. It is implemented and discussed in detail with examples. Our main contribution is a new approach which incorporates all available trust and distrust information in such a way that basic trust properties are satisfied.

Key words: Trust network, local trust values, trust properties, trust and distrust propagation, connection probability, Monte Carlo method

1 Introduction

Today, many different networks shape the Internet and serve as platforms for various kinds of interaction between unfamiliar people and businesses. The best-known examples of such platforms include eBay, Amazon, Facebook, MySpace and LinkedIn. Due to their open nature, trust between users is essential to the successful continuation of these networks. Therefore, most of them provide their own rating systems which allow users to assign trust ratings as an expression of their direct trustworthiness to other people.

Knowing this kind of information can be very useful for aggregating, filtering, and ordering data in many domains [1]. Furthermore, the consideration of trust

relationships between users is becoming a new trend for recommender systems. Trust information can be used as part of the recommendation process, especially as a means to reduce the sparsity problem [2] and thus improve the quality of recommendations [3].

However, in a community with dozens of millions of users, direct trust statements are only made to a limited subset of users. For example, on Facebook with 350 million active members and with a very simple rating system, the average user has about 130 friends [4]. This may be interpreted as 130 explicit trust statements. Because of missing direct ratings among the majority of users, there is a huge lack of trust information in most online communities. Thus, the question arises whether the existing indirect trust ratings between two unacquainted people could be used to infer and predict trust between them.

In this paper, we deal with this question and present a new approach that estimates the local trust value between any two users in a network using a Monte Carlo simulation method. The contributions of our work are as follows: At first, we propose an approach, called *MoCaTrust*, for computing trust inferences mentioned above. Thereby, we consider networks with rating systems where only positive trust statements from a specific range can be made. This reflects the situation considered by many research works in this area [5, 6]. However, experience with real trust systems like eBay and Epinions shows that distrust is at least as important as trust [7]. Thus, we take our approach a step further and present a new algorithm, called *SimTrust*, which incorporates distrust in the estimation process. To our knowledge, there are only a few works that consider distrust in the computation of local trust values. [7] use a different approach incorporating various matrix operations to evaluate atomic propagations of trust. [8] present the Moletrust algorithm for computing trust values between two users. It also allows distrust statements but ignores most of them in the calculation.

The remainder of the paper is organized as follows: In Section 2, we discuss issues like trust definition, trust properties, and trust elicitation. Section 3 describes our MoCaTrust approach and takes a look at some example calculations. In Section 4, we extend our approach and describe the SimTrust algorithm which incorporates distrust. Section 5 provides details for an implementation and points out opportunities for further optimization. The conclusion and opportunities for future work are presented in Section 6.

2 Background

In this paper, we consider the following situation: There is an online platform where users interact with each other with respect to specific interests. They assign trust values according to the platform-specific rating system. All assigned trust statements build a *trust network* represented as a directed weighted graph (see Figure 1). Users are shown as nodes and a directed edge between nodes indicates that one user (truster) made trust statement about another user (trustee). The corresponding trust value conveys how much this user trusts the other one. All trust statements are given about the same context and range from 0 to 1.

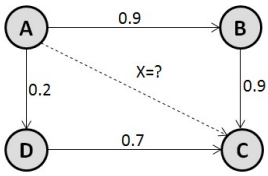


Fig. 1. A sample trust network.

The prior knowledge and experiences of the truster form the basis for trusting decisions [9]. In general, many of other factors exist which influence the decision about the trustworthiness of a user - such as subjective opinions of the actions the person has taken, recommendations from friends, psychological factors, rumors, etc. Marsh [10], for example, analyzes these factors and then, based on them, formalizes and computes trust values. This issue is a complex field and is not the scope of our work. We rather take the ratings as they exist and use them to estimate trust between two people where direct trust values are not available. However, to compute with trust, it is important to be aware of its notion and properties, the rating system for assigning trust values, and their interpretation.

2.1 Notion of trust

Considering the notion of trust, there is a variety of different understandings of its meaning in the scientific literature. Since trust is used in various ways in different disciplines, there is no standard interdisciplinary definition. In particular, social trust is a diffuse term, its properties are fuzzy, and it is not yet well understood from a computational perspective [1]. For this reason, beside the studies of trust in sociological and psychological theories [12–16] it has recently become an emerging research topic in computer science [2, 5, 7, 8, 17–19]. Here we make use of existing research results.

Sztompka, in [12], understands trust as follows: "*Trust is a bet about the future contingent actions of others*". Golbeck, in [11], adopts this definition and sees trust in a person as "*a commitment to an action based on a belief that the future actions of that person will lead to a good outcome*". Thus, trust seems to express the subjective expectation that the trusted person will behave in an appropriate manner.

Proceeding from that, trust has been identified with a subjective "*probability that [the trustee] will perform an action that is beneficial or at least not detrimental to [the truster]*" [20]. However, opinions diverge on the question of whether trust can be modeled as a mathematical probability. While some authors reject the notion of trust as a subjective probability [17, 21], others use a probabilistic interpretation [1, 6, 20, 22–24]. In our work, we also agree with [20] and assume that "*trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an [user] assesses that another [user] [. . .] will perform a particular action, both before he can monitor such action [. . .] and in a context in which it affects his own action*".

2.2 Properties of trust

Next, we deal with properties that are important for making trust computations. Based on [5], we consider three main properties of trust that hold in trust networks. The first very simple and inherent property of trust is *asymmetry*. It means, if A trusts B at a certain level, it does not necessarily mean that B trusts A at the same level.

The next important property is *transitivity*, i.e. trust can be passed along a path of trusting users. If A trusts B and B trusts C, it can be inferred that A trusts C at a certain level. However, the transitivity of trust is not perfect in the mathematical sense [5]. Considering the relationships along the path A=>B=>C in Figure 1, we cannot follow that A trusts C exactly at 0.9. On the contrary, it is plausible to expect that A's trust in C should be less than 0.9 because of their indirect relationship. Thus, trust degrades along a path of trusting people. Measuring the rate of degradation from empirical data is an interesting topic for further research.

Composability is the third property of trust. It implies that trust information along all available paths between two unknown users must be taken into account for inferring their trust. We consider again the case in Figure 1 where A trusts B and D directly. Both of them also made some trust statements about C. It is reasonable that A should take into account the information from B and D to decide about the trustworthiness of C. Another question is how this information should be composed. An analysis of the two trust networks described in [5] showed that higher trusted neighbors tend to agree with a user more than lower trusted neighbors, and that nodes connected by shorter paths tend to agree more than nodes connected by longer paths. Thus, it is reasonable that shorter and more trusted paths should have a much greater contribution to the final trust value than longer and less trusted paths.

In Section 3 and 4 we will show that MoCaTrust and SimTrust and their computational implementation satisfy all of the properties presented above.

2.3 Rating system

Another important part of a trust network is an appropriate rating system. As mentioned above, we consider two different schemes for representing trust. The first rating system, used by MoCaTrust, allows users to assign only positive trust values from the real interval (0,1) (see Section 3). In this case, e.g. the trust value of 0.1 means very low but positive trust. An expression of distrust is not possible. In our extended approach SimTrust, users also assign trust values from the interval (0,1) with the difference that trust values close to 0 mean very strong distrust, and values close to 1 represent very strong trust (see Section 4). Here, trust is seen as a point located on a probabilistic distribution of more general expectations, which can take a number of values between 0 and 1 exclusively, and which is centered around a trust threshold point T [20]. This trust threshold of e.g. 0.5 therefore stands for a neutral evaluation, with expressed distrust below this threshold and expressed trust above the threshold.

3 Computing trust using the probabilistic interpretation of trust

After the presentation of the background, we return to the main question of this paper: How much should one user trust another one if direct trust statements, based on the prior experience between them, are not available?

As mentioned above, we first assume that a trust network provides a very simple rating system according to which users can make only positive trust statements. Furthermore, the trust value assigned by a truster represents a subjective probability that the trusted person will behave in an expected manner. Each of these values is between 0 and 1 exclusively.

The idea of our approach is to interpret a trust value as the *connection probability* between two appropriate nodes, i.e. the probability that the start node connects to the target node in a network. The connection probability between nodes connected directly is given by the trust statement made by the truster. If there is no direct trust value between two nodes available, we can compute it as the *total connection probability*. It describes the probability that there is a path between these nodes in the network [25], i.e. that there exists an indirect connection between them. For finding this probability, we rely on the existing concepts and techniques which have proved their effectiveness in solving similar problems in the network reliability [26–28].

3.1 Underlying concept

The simple way to compute the total connection probability is to use the complete enumeration of all possible states of the network [26]. Consider the trust network modeled as a directed weighted graph $G = (V, E, \bar{p})$ with K nodes and M edges. Users are represented as a node set $V = \{v_1, v_2, \ldots, v_K\}$ and trust statements as an edge set $E = \{e_1, e_2, \ldots, e_M\}$. The appropriate trust values are given as $\bar{p} = (p(e_1), p(e_2), \ldots, p(e_M))$. Let $\bar{y} = (y_1, y_2, \ldots, y_M)$ denote a state of the network where $y_i = 1$ if edge e_i operates, and $y_i = 0$ if edge e_i fails, $e_i \in E$. Let Y denote the set of all the possible states of the network and

$$P(\bar{y}, \bar{p}) = \prod_{i \in E} p_i^{y_i} (1 - p_i)^{1 - y_i} \ . \tag{1}$$

Then the quantity

$$Pr(v_i, v_j) = \sum_{\bar{y} \in Y} \phi(\bar{y}) P(\bar{y}, \bar{p}) \ . \tag{2}$$

is the probability that v_i connects v_j with $v_i, v_j \in V$. The function $\phi(\bar{y}) \to [0, 1]$ defined as

$$\phi(\bar{y}) = \begin{cases} 1 & \text{if } v_i \text{ connects } v_j \text{ when state } \bar{y} \text{ occurs} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

indicates whether the target node is reachable by the source node through an indirect path when a certain state occurs.

Let us consider Figure 1 and compute how much A should trust C applying the presented method. For the depicted network there are 2^4 possible states. A few of them are listed in Table 1, as well as the respective values of ϕ and P. The

Table 1. Complete enumeration for computing $Pr(A, C)$.

Network state $\bar{y}$ (AB,BC,AD,DC)	$P(\bar{y}, \bar{p})$	$\phi(\bar{y})$
$(0,0,0,0)$	$(0.9^0 \cdot 0.1^1) \cdot (0.9^0 \cdot 0.1^1) \cdot (0.2^0 \cdot 0.8^1) \cdot (0.7^0 \cdot 0.3^1) = 0.0024$	0
$(0,0,0,1)$	$(0.9^0 \cdot 0.1^1) \cdot (0.9^0 \cdot 0.1^1) \cdot (0.2^0 \cdot 0.8^1) \cdot (0.7^1 \cdot 0.3^0) = 0.0056$	0
...	...	...
$(1,1,1,0)$	$(0.9^1 \cdot 0.1^0) \cdot (0.9^1 \cdot 0.1^0) \cdot (0.2^1 \cdot 0.8^0) \cdot (0.7^0 \cdot 0.3^1) = 0.0486$	1
$(1,1,1,1)$	$(0.9^1 \cdot 0.1^0) \cdot (0.9^1 \cdot 0.1^0) \cdot (0.2^1 \cdot 0.8^0) \cdot (0.7^1 \cdot 0.3^0) = 0.1134$	1

trust value $Pr(A, C)$ calculated as in expression (2) is 0.8366. In this simple case it can also be calculated as $Pr(A, C) = 0.9 \cdot 0.9 + (1 - 0.81) \cdot 0.2 \cdot 0.7 = 0.8366$.

Now, we recall the properties of trust discussed in Section 2.2. The asymmetry of trust holds because the trust network is modeled as a directed graph. Transitivity and composability are satisfied by the calculation rules of this method. The multiplication of trust values along the path lets trust pass from the source node to the target node. Moreover, the transitivity is not perfect because all trust values range between 0 and 1. Thus, as is to be expected, the longer the path, the more trust degrades along it. Computing the total connection probability all the possible paths are aggregated and taken into account. Therefore, the local trust value captures all available information about trust between the start node and the target node. Another effect is that each additional trust path between nodes increases the overall trust value. Above all, the information from shorter and more trusted paths inherently influences the final trust value more than that from longer and less trusted paths.

Unfortunately, the computation of the local trust value between two unacquainted users by the complete enumeration method belongs to the class of $\#P$-complete problems [26, 27]. The number of all the possible network states is 2^M and it increases exponentially with the number of edges M. This method is thus not applicable for real world networks and we need a useful approximation for the trust value. Monte Carlo sampling emerges as a well-suited solution for this kind of problem. In the next section we briefly describe a Monte Carlo method [26] that approximates the trust value effectively.

3.2 Monte Carlo approximation

The basic idea of a Monte Carlo method is to obtain the required solution by multiple repetitions of random tests and by performing some statistical analysis of the obtained results. In our case, a state of the trust network is generated

randomly, and it is determined whether the start node connects to the target node in this state. An estimator for the trust value is then simply obtained by repeating this experiment independently and calculating the mean value of the function ϕ.

To create a network state $\bar{y} \in Y$, we generate a (0, 1)-uniform distributed random number for each of the M edges and determine whether the edge operates or fails. We compare the generated number with the respective trust value. The edge fails if the random number is greater than the trust value, and this edge is removed from the graph. If the random number is lower than the trust value, the edge remains intact. Next, for this randomly generated state of the network we evaluate the function $\phi(\bar{y})$. Finally, after N repetitions of the described simulation step the solution is estimated as follows:

$$\hat{Pr}(v_i, v_j) = \frac{1}{N} \sum_{i=1}^{N} \phi(\bar{y_i}) \ . \tag{4}$$

The generation of uniformly distributed random numbers and comparison of them with the connection probability of the edge ensures that after a sufficient number of simulation steps each edge operates with that probability. Thus, the obtained relative number of existing connections from the source node to the target node is the required total connection probability. The pseudo code in Figure 2 shows how to implement the described method. Note, this algorithm has long been used to solve problems in network reliability theory [26]. We call this approach of using the Monte Carlo method to trust calculation MoCaTrust.

```
1.  counter ← 0
2.  for n ← 1 to N
3.      E' ← E
4.      for all edges e ∈ E'
5.          u ← U(0, 1)
6.          if u > p(e)
7.              then remove e from E'
8.          end if
9.      end for
10.     if v1 connects v2
11.         then counter = counter + 1
12.     end if
13. end for
14. return P̂r(v1, v2) ← counter/N
```

Fig. 2. Pseudo code of the Monte Carlo simulation method.

We now apply MoCaTrust on the sample trust network depicted in Figure 1 in order to estimate how much user A should trust user C. Figure 3 shows the simulation process by plotting the estimator of the trust value in each simulation step.

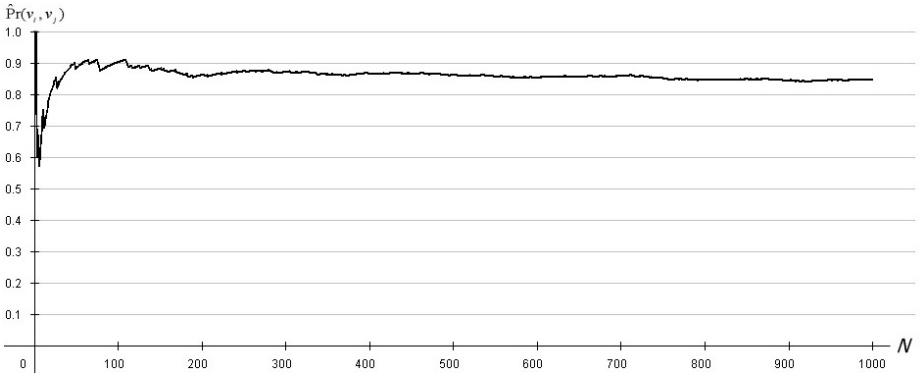


Fig. 3. Estimation of the trust value between users A and C in the sample network from Section 2 using the MoCaTrust approach.

It is easy to see that even after a few simulation steps the estimated value converges to the real trust value of 0.8366 calculated in Section 3.1 using the complete enumeration method.

4 SimTrust: Extended approach incorporating trust and distrust

4.1 Trust and Distrust

In the previous section, we introduced the MoCaTrust approach to use connection probabilities between nodes in networks to calculate local trust values. The basic approach considers only ratings where every value is treated as trust. However, most existing rating systems also include some form of distrust like a binary rating (e.g. whether a review was helpful or not) or a star ranking with four or five increments (where one star stands for 'strongly disagree').

Many existing rating systems also use distrust for trust propagation. [7] find out that distrust has significant impact on how trust is propagated and that direct distrust propagation offers the most promising results. [29] also use distrust in their models and use a similar approach. The local trust metrics of [8] and [5] also account for distrust to a certain degree. Therefore, we introduce our algorithm SimTrust, extending the simulation approach explained in Section 3 with a 'trust threshold' and transformation rules in order to account for trust and distrust.

In order to incorporate distrust, a couple of preliminary considerations have to be taken into account. Most of the existing rating systems are not able to account for a detailed and differentiated rating of a person. It is unclear whether a positive binary rating does really mean 100% trust for this person. Consider for example eBay where a vast majority of ratings is positive, and negative or even neutral feedback is hardly given. A five star ranking could be considered as a

ranking where '3' stands for indifferent or neutral and '4' and '5' are equidistant values somewhere between 0.5 and 1.0. But other interpretations are also possible which makes it difficult to work with these imprecise values for local trust values.

In this paper, we use a rating system where users assign trust values from the continuous interval (0,1). So, they have the possibility to rate each other in detail. This initial rating scale includes both trust and distrust; therefore a value close to 0 represents very strong distrust, and a value close to 1 means very strong trust. With regards to the meaning and interpretation of existing trust values, they could be transformed to this continuous scale. For example, the 5 star rating could stand for 0.1, 0.3, 0.5, 0.7 and 0.9 to work with our approach.

These continuous trust values cannot be treated as probabilities in networks directly. Intuitively, if the source user A rates the target user B with a trust value below a certain threshold, this rating should reduce the overall local trust value from user A to user B. According to [20], we call this value 'trust threshold' T. Naturally, this trust threshold would be 0.5 in most cases[1] - which is also the trust value issued to individuals where no information is available [20]. Thus, every value below 0.5 shall reduce the overall trust value, and every value above 0.5 shall increase it respectively. However, as stated earlier, when calculating trust in networks every path from the source user to the target user increases the overall trust value.

4.2 Calculating local trust values using trust and distrust

For calculation purposes, we therefore introduce two temporary graphs G^{trust} and $G^{distrust}$ that are calculated separately in order to account for trust and distrust. These two graphs are copies of the original graph $G = (V, E, \bar{p})$ where the values of the direct predecessors of the target node are modified. For these direct predecessors of the target node, it is intuitively clear that a low trust value of e.g. 0.1 should reduce the local trust value for this target and a high trust value of e.g. 0.9 improves the local trust value.

For the temporary trust graph, we therefore use the trust threshold T and transform the trust values above this threshold to a new scale ranging from 0 to 1 using a simple linear transformation. The incoming edges of the target node with values below the threshold ($p(e_i) < T$) are deleted (setting the probability to zero). Hence, in the temporary trust graph $G^{trust} = (V, E', \bar{p'})$, every value $p(e_i)$ of direct predecessors of the target node is transformed according to:

$$p'(e_i) = \begin{cases} (p(e_i) - T) \cdot \frac{1}{1-T} & \text{if } p(e_i) \geq T \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

For a trust threshold of $T = 0.5$ this leads to

$$p'(e_i) = \begin{cases} (p(e_i) - 0.5) \cdot 2 & \text{if } p(e_i) \geq 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

[1] For certain networks and with respect to real data, other values than 0.5 are possible for T, for example in networks with overall positive ratings the threshold could be set other values like 0.3, 0.7 or 0.9 in order to capture the average or median overall value.

Correspondingly, in the temporary distrust graph $G^{distrust} = (V, E'', \bar{p''})$, every value $p(e_i)$ of direct predecessors of the target node is transformed according to:

$$p''(e_i) = \begin{cases} (T - p(e_i)) \cdot \frac{1}{T} & \text{if } p(e_i) < T \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

And for $T = 0.5$:

$$p''(e_i) = \begin{cases} (0.5 - p(e_i)) \cdot 2 & \text{if } p(e_i) < 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

We only change values of direct predecessors of the target node and do not change the values 'in the middle'. The reasons for not changing these values are given below.

There are many approaches to treat these values. [7] introduce and evaluate some of them. One extreme suggests that the "*enemy of your enemy is your friend*" [30] and these values should therefore be inverted. However, such interpretation only holds true in very special circumstances. Other approaches like Moletrust [8] suggest ignoring all values below a certain threshold completely and deleting the corresponding edges. This approach might lead to a significant reduction of edges in the graph but at the same time significant distrust information might get lost.

We, on the other hand, use the interpretation that even a strong distrust towards a node does not necessarily mean that the judgment of this node is flawed when rating others. For example, a person writing low quality reviews in a product portal might very well be able to rate other reviews. Thus, it is reasonable to consider such ratings to a certain degree. Of course, trust values issued from distrusted nodes should have a much lower effect than those issued by edges where we have a strong trust connection. This property holds for SimTrust if no modification is done to the trust values in the middle at all. You can see that a path including a value like 0.1, indicating strong distrust towards a person, has only a minor effect on the overall local trust values, especially when other trusted paths suggest strong trust or distrust.

It can also be shown that our approach calculates much better values if a trust value along a path is just slightly below the threshold value used for deleting edges in other approaches. E.g., in [8], changing one trust value just slightly can have significant changes on the overall local trust value if this change moves the value above or below the threshold and there are few paths from source to target. Our approach avoids this harsh cutoff depending on a binary segmentation in trust (which counts fully) and distrust (which is ignored).

For each of the two temporary graphs G^{trust} and $G^{distrust}$ with the transformed values of the direct predecessors of the target nodes, we now run the simulation described in Section 3. This leads to two trust values $\hat{Pr}^{trust}(v_i, v_j)$ and $\hat{Pr}^{distrust}(v_i, v_j)$. These values are now inversely transformed to get the overall trust value:

$$\hat{Pr}^{local}(v_i, v_j) = \begin{cases} D \cdot T + T & \text{if } D < 0 \\ D \cdot (1 - T) + T & \text{if } D \geq 0 \end{cases} \tag{9}$$

where $D = \hat{Pr}^{trust}(v_i, v_j) - \hat{Pr}^{distrust}(v_i, v_j)$.

This transformation brings the two values back to the original scale where T is the trust threshold between trust and distrust. The first case is used if the trust value is higher than the distrust value. The overall trust value then lies between T and 1. In the second case, distrust outweighs trust and the overall trust value falls between 0 and T. For a threshold of $T = 0.5$ the equation is simplified to:

$$\hat{Pr}^{local}(v_i, v_j) = \frac{\hat{Pr}^{trust}(v_i, v_j) - \hat{Pr}^{distrust}(v_i, v_j)}{2} + 0.5\ . \tag{10}$$

An implementation of the complete algorithm including procedure, evaluation and results can be found in Section 5.

5 Implementation

In order to evaluate SimTrust, we have implemented the approach in a simple program. The basic procedure of the approach (as explained in detail in Section 4) can be found in Table 2.

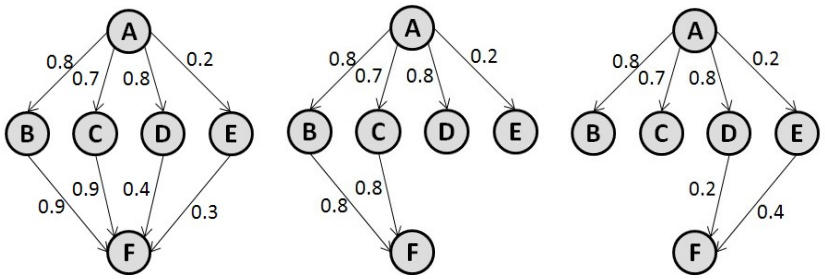
Table 2. Procedure of SimTrust algorithm.

1	Duplicate the original graph twice into G^{trust} and $G^{distrust}$
2	Find the direct predecessors of the target node
3	Transform both graphs according to the rules in equation (5) and (7)
4	Run simulation on G^{trust} and save obtained value in $\hat{Pr}^{trust}(v_i, v_j)$
5	Run simulation on $G^{distrust}$ and save obtained value in $\hat{Pr}^{distrust}(v_i, v_j)$
6	Reverse transform the obtained values using equation (9)
7	Display the output local trust value

We use JUNG 'Java Universal Network/Graph Framework' [31] as a basis for our implementation. The implementation itself is quite straightforward using mostly methods already implemented by JUNG. We use the Dijkstra algorithm [33] in order to determine the reachability of target and source nodes during the simulation steps.

5.1 Example

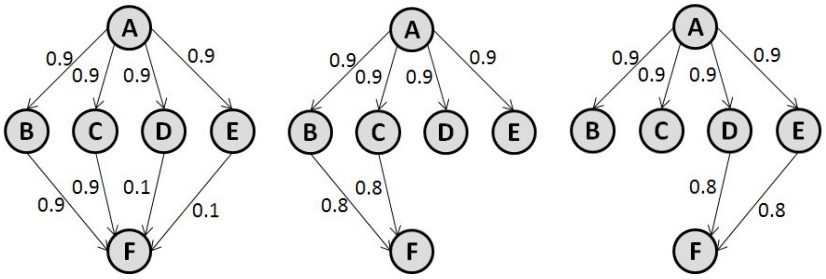
We now show an example calculation of the complete SimTrust algorithm using two sample graphs. Figure 4 shows a sample graph and the two transformations to G^{trust} and $G^{distrust}$ for the trust threshold $T = 0.5$. The two simulation runs converge to $\hat{Pr}^{trust}(A, F) = 0.8416$ and $\hat{Pr}^{distrust}(A, F) = 0.2272$. Inverse transformation leads to the overall trust values of $\hat{Pr}^{local}(A, F) = 0.8072$.



(a) Original graph (b) G^{trust} (c) $G^{distrust}$

Fig. 4. A sample graph and transformations for $T = 0.5$.

Figure 5 presents another example where we have two positive and two negative ratings of the target node. The source node A rates the nodes B, C, D, and E with an equal trust value of 0.9. Further, B and C rate the target node F with strong trust while D and E express strong distrust. Intuitively, based on this information neither trust nor distrust can be predicted.



(a) Original graph (b) G^{trust} (c) $G^{distrust}$

Fig. 5. A sample graph with neutral trust prediction.

In fact, the SimTrust algorithm results in a neutral trust value: The simulation runs converge to $\hat{Pr}^{trust}(A, F) = 0.9216$ and $\hat{Pr}^{distrust}(A, F) = 0.9216$. The overall trust calculated as in expression (10) is $\hat{Pr}^{local}(A, F) = 0.5$.

5.2 Evaluation

Complexity. In this section, we will evaluate the complexity of our approach using the Big-O notation; for an introduction, see e.g. [32]. According to graph theory literature, n stands for the number of nodes and m stands for the number of edges in the current graph. Duplicating the graph has a complexity of $O(1)$. Transforming the graph has a worst case complexity of $O(m)$ if all users rate the target node.

The simulation runs are the most complex parts. Here, checking the reachability (whether the source node connects the target node) is most complex. We use the Dijkstra algorithm [33] in our implementation which, using some optimization techniques, has a complexity of $O(m + n \cdot log(n))$ if all distances are positive and there are less than n^2 edges [34]. Since we do not need to find the shortest path but are only concerned with reachability, there might be more efficient ways to solve the problem. [35] introduce an algorithm with almost linear update time. Among others, [36, 37] also introduce reachability algorithms that are more efficient than Dijkstra. Since all of them have specific restrictions or limitations, an optimal algorithm depends on the specifics of actual networks. The rest of the SimTrust algorithm works in constant time. Reverse transformation and output both work in $O(1)$.

In summary, the current implementation has a complexity of $O(m+n \cdot log(n))$. Note that in complexity theory, only the most complex part is used for the total complexity. A constant factor of 1000 simulation steps also does have no effect on the complexity. Approaches to improve the current implementation are given in the next section.

Optimization. As already stated above, our current implementation is only a proof of concept with a few suggestions for efficient algorithms. For an implementation in a real world scenario, there are several possibilities to improve our approach. Some of them are presented below.

First, only paths that have a certain maximum depth could be used. Due to the multiplication of trust values, it can be shown that starting from a certain path length, additional values have little to no effect. The exact depth has to be evaluated depending on real implementations.

Second, our current approach uses a fixed number of simulation steps. In order to improve the runtime, a dynamic number of simulation steps can be used. For this purpose, however, an appropriate convergence diagnostic is needed.

Finally, because of the parallel nature of the SimTrust algorithm (each simulation step does not depend on the result of the previous one), it can be implemented in a parallel environment. Furthermore, if local trust values for more than one target user shall be calculated at the same time, distributing the code between various (virtual) computers might also improve the runtime.

6 Conclusions and future work

In most online platforms, trust is a prerequisite for successful interaction. The a priori knowledge of how much a user should trust other unfamiliar users can considerably improve its quality. In this paper, we presented two approaches to infer local trust values between unknown users based on their indirect trust statements. We first considered the situation where only positive trust statements could be made and introduced MoCaTrust, an approach for calculating these trust values using a Monte Carlo simulation method. Next, we modified it and presented the SimTrust algorithm in order to account for trust and distrust. Implementation details and examples were given for illustration.

In the future, our intention is to improve the described approach, particularly with regard to the runtime, and evaluate it on real world networks. The latter could not be done yet because of the unavailability of suitable real trust and distrust data.

References

1. Kuter, U., Golbeck, J.: SUNNY: a new algorithm for trust inference in social networks using probabilistic confidence models. In: Proceedings of the 22nd national conference on Artificial intelligence, pp. 1377–1382. AAAI Press (2007)
2. O'Donovan, J., Smyth, B.: Trust in recommender systems. In: Proceedings of the 10th International Conference on Intelligent User Interfaces, pp. 167–174. ACM, San Diego, USA (2005)
3. Avesani, P., Massa, P., Tiella, R.: A trust-enhanced recommender system application: Moleskiing. In: Proceedings of the 2005 ACM symposium on Applied computing, pp. 1589–1593. ACM, Santa Fe, New Mexico (2005)
4. Facebook, http://www.facebook.com/press/info.php?statistics
5. Golbeck, J.A.: Computing and applying trust in web-based social networks, University of Maryland at College Park (2005)
6. Richardson, M., Agrawal, R., Domingos, P.: Trust Management for the Semantic Web. In: Proceedings of the Second International Semantic Web Conference, pp. 351–368 (2003)
7. Guha, R., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of trust and distrust. In: Proceedings of the 13th international conference on World Wide Web, pp. 403–412. ACM, New York, USA (2004)
8. Massa, P., Avesani, P.: Controversial users demand local trust metrics: an experimental study on Epinions.com community. In: Proceedings of the 20th national conference on Artificial intelligence, pp. 121–126. AAAI Press, Pittsburgh, Pennsylvania (2005)
9. Luhmann, N.: Trust and Power. John Wiley & Sons Inc., Chichester (1979)
10. Marsh, S.P.: Formalising Trust as a Computational Concept, University of Stirling, Department of Mathematics and Computer Science (1994)
11. Golbeck, J.A.: Trust and nuanced profile similarity in online social networks. ACM Trans. Web, vol. 3, pp. 1–33 (2009)
12. Sztompka, P.: Trust: A Sociological Theory. Cambridge University Press (2000)
13. Luhmann, N.: Familiarity, Confidence, Trust: Problems and Alternatives. In: Gambetta, D. (ed.) Trust: Making and Breaking Cooperative Relations, pp. 94–107. University of Oxford (2000)
14. Coleman, J.: Foundations of Social Theory. The Belknap Press of Harvard University Press (1990)
15. Falcone, R., Castelfranchi, C.: Social trust: a cognitive approach. In: Castelfranchi, C. and Tan, Y. (eds.) Trust and deception in virtual societies, pp. 55–90. Kluwer Academic Publishers (2001)
16. Williamson, O.: Calculativeness, Trust, and Economic Organization. Journal of Law and Economics, vol. 36, pp. 453–486 (1993)
17. Abdul-Rahman, A., Hailes, S.: Supporting Trust in Virtual Communities. In: Abdul-Rahman, A. and Hailes, S. (eds.) HICSS 2000. Proceedings of the 33rd Hawaii International Conference on System Sciences (2000)

18. Massa, P., Bhattacharjee, B.: Using Trust in Recommender Systems: An Experimental Analysis. In: Trust Management: Second International Conference, iTrust 2004, pp. 221–235 (2004)
19. Golbeck, J.A. ed: Computing with Social Trust. Springer, Berlin (2008)
20. Gambetta, D.: Can We Trust Trust? In: Gambetta, D. (ed.) Trust: Making and Breaking Cooperative Relations, pp. 213–237. University of Oxford (2000)
21. Castelfranchi, C., Falcone, R.: Trust Is Much More than Subjective Probability: Mental Components and Sources of Trust. In: Proceedings of the 33rd Hawaii International Conference on System Sciences. IEEE Computer Society (2000)
22. Dasgupta, P.: Trust as a Commodity. In: Gambetta, D. (ed.) Trust: Making and Breaking Cooperative Relations, pp. 49–72. University of Oxford (2000)
23. DuBois, T., Golbeck, J., Srinivasan, A.: Rigorous Probabilistic Trust-Inference with Applications to Clustering. In: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, pp. 655–658. IEEE Computer Society (2009)
24. Gulati, R.: Does Familiarity Breed Trust? The Implications of Repeated Ties for Contractual Choice in Alliances. The Academy of Management Journal, vol. 38, pp. 85–112 (1995)
25. Chen, Y., Li, J., Chen, J.: A new algorithm for network probabilistic connectivity. In: Military Communications Conference Proceedings, pp. 920–923 (1999)
26. Fishman, G.: Monte Carlo: Concepts, Algorithms, and Applications. Springer, Berlin (1996)
27. Provan, J.S., Ball, M.O.: The Complexity of Counting Cuts and of Computing the Probability that a Graph is Connected. SIAM Journal on Computing, vol. 12, pp. 777–788 (1983)
28. Agrawal, A., Satyanarayana, A.: An O(|E|) Time Algorithm for Computing the Reliability of a Class of Directed Networks. Operations research, vol. 32, pp. 493–515 (1984)
29. Ziegler, C., Lausen, G.: Propagation Models for Trust and Distrust in Social Networks. Information Systems Frontiers, vol. 7, pp. 337–358 (2005)
30. Rosenberg, J.L.: The 1941 Mission of Frank Aiken to the United States: An American Perspective. Irish Historical Studies, vol. 22, pp. 162–177 (1980)
31. JUNG, http://jung.sourceforge.net/index.html
32. Sipser, M.: Introduction to the Theory of Computation. International Thomson Publishing (1996)
33. Cormen, T.H., Stein, C., Leiserson, C.E., Rivest , R.L.: Introduction to Algorithms. B & T (2001)
34. Goldberg, A.V., Tarjan, R.E.: Expected Performance of Dijkstra's Shortest Path Algorithm, NEC Research Institute Report (1996)
35. Roditty, L., Zwick, U.: A fully dynamic reachability algorithm for directed graphs with an almost linear update time. In: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, pp. 184–191. ACM, Chicago, IL, USA (2004)
36. Khoussainov, B., Liu, J., Khaliq, I.: A Dynamic Algorithm for Reachability Games Played on Trees. In: Mathematical Foundations of Computer Science 2009. LNCS, vol. 5734, pp. 477–488. Springer, Berlin / Heidelberg (2009)
37. Wang, H., He, H., Yang, J., Yu, P.S., Yu, J.X.: Dual Labeling: Answering Graph Reachability Queries in Constant Time. In: Proceedings of the 22nd International Conference on Data Engineering (ICDE'06), pp. 75–87. IEEE Computer Society (2006)