Improving the Effectiveness of Eigentrust in Computing the Reputation of Social Agents in Presence of Collusion

This is a postprint version (accepted version) of the paper published in International Journal of Neural Systems, Vol. 34, No. 02, 2350063 (2024), available at https://www.worldscientific.com/doi/10.1142/S0129065723500636

IMPROVING THE EFFECTIVENESS OF EIGENTRUST IN COMPUTING THE REPUTATION OF SOCIAL AGENTS IN PRESENCE OF COLLUSION*

MARIANTONIA COTRONEI
Department of Information Engineering, Infrastructure and Sustainable Energy (DIIES), University Mediterranea of Reggio Calabria, Feo di Vito, Reggio Calabria, 89122, Italy
E-mail: mariantonia.cotronei@unirc.it
www.unirc.it

SOFIA GIUFFRÈ
Department of Information Engineering, Infrastructure and Sustainable Energy (DIIES), University Mediterranea of Reggio Calabria, Feo di Vito, Reggio Calabria, 89122, Italy
E-mail: sofia.giuffre@unirc.it
www.unirc.it

ATTILIO MARCIANÒ
Department of Information Engineering, Infrastructure and Sustainable Energy (DIIES), University Mediterranea of Reggio Calabria, Feo di Vito, Reggio Calabria, 89122, Italy
E-mail: attilio.marciano@unirc.it
www.unirc.it

DOMENICO ROSACI
Department of Information Engineering, Infrastructure and Sustainable Energy (DIIES), University Mediterranea of Reggio Calabria, Feo di Vito, Reggio Calabria, 89122, Italy
E-mail: domenico.rosaci@unirc.it
www.unirc.it

GIUSEPPE M. L. SARNÈ†
Department of Psychology, University of Milan Bicocca, Piazza dell'Ateneo Nuovo, 1, Milano, 20126, Italy
E-mail: giuseppe.sarne@unimib.it
www.unimib.it

The introduction of trust-based approaches in social scenarios modeled as multi-agent systems (MAS) has been recognized as a valid solution to improve the effectiveness of these communities. In fact, they make interactions taking place in social scenarios much fruitful as possible, limiting or even avoiding malicious or fraudulent behaviors, including collusion. This is also the case of multi-layered neural networks (NN), which can face limited, incomplete, misleading, controversial or noisy datasets, produced

*An early version of this paper has been presented in[15]
†Corresponding author

by untrustworthy agents. Many strategies to deal with malicious agents in social networks have been proposed in the literature. One of the most effective is represented by Eigentrust, often adopted as a benchmark. It can be seen as a variation of PageRank, an algorithm for determining result rankings used by search engines like Google. Moreover, Eigentrust can also be viewed as a linear neural network whose architecture is represented by the graph of Web pages. A major drawback of Eigentrust is that it uses some additional information about agents that can be a priori considered particularly trustworthy, rewarding them in terms of reputation, while the non pre-trusted agents are penalized. In this paper, we propose a different strategy to detect malicious agents which does not modify the real reputation values of the honest ones. We introduce a measure of effectiveness when computing reputation in presence of malicious agents. Moreover, we define a metric of error useful to quantitatively determine how much an algorithm for the identification of malicious agents modifies the reputation scores of the honest ones. We have performed an experimental campaign of mathematical simulations on a dynamic multi-agent environment. The obtained results show that our method is more effective than Eigentrust in determining reputation values, presenting an error which is about a thousand times lower than the error produced by Eigentrust on medium-sized social networks.

Keywords: Trust; Reputation; Social Networks; Recursive Models; Multi-Agent Systems.

1. Introduction

A particularly dangerous type of fraud in modern social networks is represented by *collusion*. Collusion consists in a deceptive agreement or secret cooperation between two or more agents to restrict free competition by cheating, misleading or defrauding others of their legal right.

Many strategies to deal with malicious agents in social networks have been proposed in the literature and in Section 2 we briefly overview some of them.

More in general, among these proposals, one of the most effective is represented by PageRank,[59] the well-known algorithm used by the Google search engine for determining result rankings. In PageRank, the importance (reputation) of a web page is measured, in particular, according to the number of links to that specific web page.

It is possible to represent the World Wide Web as a directed graph, in which Web pages are nodes and an edge from a page $P1$ to a page $P2$ means that there is an hyperlink to $P2$ on $P1$. From the description of the PageRank algorithm, we can see that the probability of going from $P1$ to $P2$ is described exactly by the formula used for computing the weights a_{P1P2} of a linear neural network. In this way, the final probabilities, i.e. the ranks of the different pages, form an eigenvector of the corresponding matrix of the weights. Thus, PageRank can be viewed as a natural application of linear neural networks[45]. Another example of application of the PageRank algorithm, seen as linear neural networks, is to provide an effective ranking of papers by their importance, where the importance is based on the citations obtained by the papers. This idea has been proposed already by the authors of the PageRank algorithm.

An important alternative to PageRank is Eigentrust, which has the primary focus on determining sureness and authenticity of traffic in P2P Networks. Eigentrust can be effectively used to detect malicious agents in a social agent network. In particular, in Eigentrust, a *trust value* t_{ij} is assigned from each agent i to each other agent j, while a *reputation* r_i is assigned from the whole community to each agent i. Instead, trust is a subjective measure, evaluated by a trustor about a trustee. In this respect, the reputation of an agent i, is computed as a weighted mean of all the opinions about i provided by the other agents. Therefore, reputation has to be considered as a synthetic evaluation about the opinion that the whole community, and not only a single agent, has with regard to i. Thanks to its effectiveness, Eigentrust is also one of the most highly considered benchmarks to evaluate/train other systems identifying malicious agents.

We highlight that Eigentrust, in addition to being equivalent to linear neural networks, can be usefully applied also to make the training activity of any neural network[14] more effective and efficient. Indeed, often a neural network has to be trained with data coming from unreliable or even malicious sources. The aforementioned case of providing rankings of scientific papers based on their importance is an exam-

ple of such a type of situation. In fact, since some authors may adopt unethical behavior in using citations, for example by flattering themselves in citing each other (collusion). Eigentrust could be fruitfully exploited to identify such types of malicious sources, eliminating them from the training set or reducing their impact on the training.

However, two important issues should be highlighted to apply this algorithm, namely:

(1) Eigentrust needs to receive additional information about agents to consider a-priori particularly trustworthy as inputs. The task to determine such agents is not trivial in the case of collusion.
(2) The application of Eigentrust to a social agent community presents the side-effect of artificially modifying the reputation values of the non colluded agents. This is not a desired effect when reputation analysis is a main target of the system, besides of the detection of the colluded agents. Indeed, we highlight that the final goal of any reputation mechanism in presence of malicious agents is not only to detect malicious agents. In fact, a main goal is also to determine the reputation values that the agents would assume in absence of malicious agents in the system. These values can be considered as "ideal reputation values". Eigentrust, although it is very efficient to detect malicious agents, adopts a strategy that leads to assign to the honest agents reputation values that are relatively lower than their ideal reputation values.

While trust is a subjective measure, evaluated by a trustor with respect to a trustee (in a peer-to-peer interaction), *reputation* is instead an evaluation about the trustworthiness that the whole community has with regard to a given agent. In a nutshell, the Eigentrust algorithm uses some additional information about agents that can be a-priori considered particularly trustworthy (i.e. the pre-trusted agents), rewarding them in terms of reputation, while the non pre-trusted agents are penalized. It applies its rewarding operation to all the pre-trusted agents, regardless of their reputation values. The method is thus capable to effectively detect malicious agents (that are identified as the worst reputed agents among the non pre-trusted agents). Unfortunately, Eigentrust generates the side effect to artificially modify the reputation of the other agents, which are all indiscriminately awarded. As a consequence, the differences, in terms of reliability, between honest agents are flattened. We highlight that, since the reputation values are upper bounded by the value 1, the operation of indiscriminately rewarding all the honest agents tends to increase the reputation values of these latter towards the upper bound, thus flattening the differences between those agents.

Moreover, Eigentrust needs to yield as input the information about pre-trusted agents, that in the case of collusion is not trivial to be found.

Given the observations above, in this paper we provide the following contributions. First, we propose a simple strategy which allows to preliminary detect the best candidates as malicious (colluding) agents directly from the matrix $T = [t_{ij}]$. We use these agents as pre-untrusted agents. Second, we introduce a different strategy to detect the malicious agents with respect to that used by Eigentrust. In other words, instead of indiscriminately rewarding pre-trusted agents, artificially decreases the fraudulent trust values that colluding agents mutually exchange. This way, besides of detecting the malicious agents with the same effectiveness of Eigentrust, we do not modify the real reputation values of the honest agents. Finally, we introduce a measure of effectiveness in computing reputation in presence of malicious agents. The consequential metric of error is useful to quantitatively determine how much an algorithm for the identification of malicious agents modifies the correct reputation values of the honest agents. We have used such a metric to compare the effectiveness of our result with that generated by Eigentrust.

We have performed an experimental campaign of mathematical simulations on a dynamic multi-agent environment, showing that our method is more effective than Eigentrust in determining reputation values. Our method presents, an error, defined as above, which is about a thousand times lower than the error produced by Eigentrust on medium-sized social networks. It is important to highlight that our experiments are exclusively devoted to improving the effectiveness of the methodology used by Eigentrust in terms of precision of the computed reputation values. In fact, we are not trying to improve the capability of Eigentrust to detect malicious agents. In particular, the robustness of a reputation system is not

an absolute property, but depends on several aspects such as model characteristics, application scenario, and class of attack. Although context changes may not be performance-neutral, benchmarks on major reputation systems confirm that, in the considered context, Eigentrust is the best performing approach against collusion attacks[9]. This is also true for the presence of pre-trusted peers, although with the side effect of ranking other peers lower despite their honesty. In any case, comparison with other reputation systems is outside our scope, which is only to minimize the aforementioned Eigentrust side effect.

The rest of the paper is organized as follows. In Section 2 we discuss some related work. In Section 3, we introduce the scenario we deal with and the mathematical foundations of the recursive reputation model which our algorithm and Eigentrust are relying on. In Section 4, we describe an example of application of the algorithms presented in the previous section, to make more understandable the difference between them. In Section 5, we propose two error measures, to make a quantitative analysis of the results. Then, in Section 6, we describe the simulations we have performed to compare our algorithm with Eigentrust. Finally, in Section 7, we draw some conclusions and discuss our ongoing research.

2. Related Work

Intelligent software agents are largely exploited in many areas for their autonomous, adaptive, learning, proactive and social capabilities (e.g., in decision support systems,[4] conversational tools,[3] hazard scenarios,[20] smart factories,[25] neural models,[34] transportation,[63] control systems[32] and many others). Also Trust and Reputation Systems (TRSs) exploit agents to simulate a variety of complex, uncertain, dynamic human and agent-to-agent behaviors at different levels of detail and abstraction.

A number of studies compared the robustness of TRSRs to a variety of misbehaviors and contexts. The earliest comparisons[36,41,72] did not perform quantitative evaluations on strategies, weaknesses, and strengths. Later, several testbeds where reliable and unreliable actors compete with each other to gain some advantage were used. Well known is ART (Agent Reputation and Trust)[27] but other remarkable testbeds are described in.[2,44,53] Usually, testbeds are unable to autonomously identify the worst conditions and may lead to errors when TRSs are moved to different contexts. In contrast, mathematical or analytical approaches[10,30] are more complex, more effective but lack of generality.

The simplicity of the eBay RS[37] made it popular, although it is exposed to many threats.[12,33,64] This RS increases or decreases reputation (starting from 0) based on feedback issued by users that also have to interpret the reputation scores. Over time, this RS has been updated to improve its resilience to malicious but without changing its basic nature.

PeerTrust[77] is a distributed agent transaction-based RS, over a peer-to-peer overlay network, to identify peers to collaborate. It considers direct peers' feedback, number and context of peer' transactions, and credibility of indirect feedback sources to improve resilience against threats. Also Hypertrust[54] adopts a distributed approach to discover and allocate trusted resources into competitive, large federations of utility infrastructures. Its metric considers reliability and reputation (from opinions) sources. A decentralized procedure builds an overlay network with all the federation nodes to implement an efficient finding process for computational resources.

Eigentrust[43] has been designed for file sharing peer-to-peer networks. Here, each peer builds its own local trust representation about the whole peer community. All the normalized peers' trust reputations, weighted by the trustworthiness of the peer, are collected in a matrix, called global reputation. The authors show how such values asymptotically converge to the matrix eigenvalues. The malicious discrimination uses an arbitrary reputation threshold. To minimize the influence of colluding activities, some peers are assumed as trusted. This RS is susceptible for peers' feedback manipulations,[39] and several changes have been introduced to increase its robustness[1,21,46] or adapt it to new scenarios.[28,49] Due to its performance, Eigentrust is a popular benchmark for evaluating other TRRs.[6,9,48]

More in detail, Eigentrust is a variation of the PageRank algorithm.[11] PageRank and Eigentrust have met the interest of NN researchers not only because PageRank (i.e., Eigentrust) is equivalent to a linear Neural Network (NN)[45] but also for their potentialities in the NN area. Some examples can be found in[69] with a graph NN that implements PageRank or in[19] where a method to integrate global graph information into supervised Word Sense Disambiguation (WSD) neural networks through a per-

sonalized PageRank approximation is described. The relationships between different NNs and PageRank are investigated in.[29,71,76] Authors of[73] exploit sets of data simulated by Eigentrust to train NNs, and in[5] Eigentrust directly supports a NN in a synergistic way, while in[6] it is used as benchmark to verify the NN training quality. Obviously, all NNs exploiting Eigentrust-based reputation scores can benefit from the increased effectiveness given by our method to also significantly improving their performances.

Sometimes TRSs are just a theoretical exercise. For example, FIRE[38] idealistically imposes benevolence (e.g., agents freely exchange information) and honesty. Its model considers more types of trust and feedback sources for robustness but benevolence and honesty assertions make FIRE vulnerable to multiple types of strikes (e.g., collusion, alternate, complaining, etc.)[68] Besides, its complex setting is another important criticism. Consequently, FIRE unfits with contexts where malicious actors operate.

In the presence of large communities, computing the reputations of all agents is not a trivial task both because of the inherent computational complexity, particularly in the presence of centralized systems,[66] and because of the frequent need to proceed by successive refinements. The latter, besides being a common strategy in a significant number of TRRs,[58,79,82] and not only,[47,56,75] is also the solution we adopted.

3. Effective reputation computation

In this section we present the recursive reputation model. Before developing the model, we introduce our notation.

Notation

n	# of agents
$T = [t_{ij}]$	$n \times n$ *original trust matrix* where t_{ij} is the trust perceived by member j with respect to the member i
$\widetilde{T}$	$n \times n$ *Eigentrust matrix*
$\widehat{T}$	$n \times n$ *matrix obtained with our strategy*
r	$n \times 1$ *reputation vector*
v	$n \times 1$ *teleportation vector*
u	$n \times 1$ vector of ones
d	$n \times 1$ row-sums vector of T
$\mathcal{P}$	Set of *pre-trusted agents*
$\mathcal{F}$	Set of *potential colluded agents*
$\mathcal{M}$	Set of *malicious agents*

3.1. *The Recursive Reputation Model*

In our scenario, we suppose to have a social network composed by n members, each member being uniquely identified by an integer i, $1 \leq i \leq n$.

The *reputation* r_i of each member i of the social network can be seen as the sum of all the trust values t_{ij}, $j = 1, \ldots, n$, corresponding to a trustor j, weighted by the reputation r_j of j. Without loss of generality, we will consider evaluations over the interval $[0, 1]$, i.e. $T = [t_{ij}] \in [0, 1]^{n \times n}$ and the reputation vector $r \in [0, 1]^n$. Moreover, we assume that each member j may distribute an overall sum of the trust values t_{ij} equal to 1, namely $\sum_{i=1}^{n} t_{ij} = 1$, then the matrix T will be column-stochastic.

The reputation r_i can also be viewed as the barycenter of all the trust values expressed for i by the trustors. Formally:

$$r_i = \frac{\sum_{j=1}^{n} t_{ij}\, r_j}{\sum_{j=1}^{n} r_j}, \quad i = 1, \ldots, n. \tag{1}$$

It is worthwhile to remark that it can be seen as the transpose of the weighted adjacency matrix $A = [a_{ij}]$ of a directed graph $\mathcal{G}$ where each node is associated with a user and a non negative value a_{ij} is assigned to the edge (i, j), representing the trust score assigned by the member i to the member j.

The *reputation vector* $r = (r_1, \ldots, r_n)^T$, whose elements satisfy (1), can be computed as solution to the system of linear equations:

$$Tr = r \tag{2}$$

Since the matrix T is column-stochastic (equivalently it is a transition or a Markov matrix), as it is well-known, if we further assume that its entries are positive, the solution to (2) is guaranteed by the Perron-Frobenius theorem. In particular, it follows that $\lambda = 1 = \rho(T)$ is a simple eigenvalue of T (the other ones being in modulus less than 1), and there exists a corresponding unique eigenvector $r \in \mathbb{R}^n$, $\|r\|_1 = 1$, that is a unique positive reputation vector whose elements sum up to 1.

This vector (also called the steady-state vector) represents the stable equilibrium distribution of the Markov chain represented by the transition matrix T.

A modified version of the eigensystem (2) is represented by the well-known *PageRank model*:

$$\widetilde{T}r = r \tag{3}$$

where

$$\widetilde{T} = \alpha T + (1-\alpha)vu^T$$

represents the new trust matrix, $0 \leq \alpha \leq 1$, u is the vector of all ones, v is a non-negative vector with unitary 1-norm, that is $u^T v = 1$.

The vector v is usually called *teleportation vector*.

PageRank's original algorithm adopts the uniform choice $v = \frac{1}{n}u$. In such a case, when $\alpha \neq 0, 1$, the existence and uniqueness of the solution to (3) are guaranteed.

In the 2003 paper by Kamvar et al. [43] a modified model is proposed, known as *Eigentrust algorithm*. Its *basic* version consists in solving the same system as in (3), by choosing v in a way useful to mitigate the final reputation of malicious users. In particular, by denoting with $\mathcal{P}$ the set of the so-called *pre-trusted* agents, i.e. agents that can be a-priori considered particularly trustworthy, the entry v_i of the vector v is equal to $1/|\mathcal{P}|$, if the i-th user belongs to $\mathcal{P}$, 0 otherwise.

In this approach, the pre-trusted users are known a priori. They coincide, for example, with the first few members to join the network.

The detection of potential malicious users, on the other hand, could be better exploited for a more realistic computation of the reputation vector. In the next subsection, we propose a strategy that allows to "read" such information straight from the matrix T.

3.2. *Detection of potential malicious users*

We aim at investigating collusion in social networks, so we first provide a definition of malicious agents and, then, a threshold strategy for concretely detecting them.

We stress that by means of this approach we detect the best candidates as malicious agents directly from the matrix T.

Once the malicious are identified, we can use this information in two ways: either considering the non-colluding agents as pre-trusted in the Eigentrust model (3) or employing an alternative algorithm which specifically makes use of such data.

We classify a pair of agents i, j as *malicious* if they *collude*, i.e. when the two conditions listed below are verified at the same time:

- the trusts t_{ij} and t_{ji} take high values (and they are similar to each other);
- the sum of the residual trust scores belonging to the rows i and j takes a low value.

We thus check in the matrix T for high values t_{ij} corresponding to high values t_{ji}, associated with users which receive low values from the most part of the members.

We observe that the collusion relationship is not transitive, since the collusion between two agents A and B and the collusion between the agents B and C do not necessarily imply a collusion between A and C, and the transitive property is not requested by our model.

For a proper mathematical formulation of the above procedure, we introduce the vector d of the node out-degrees of the graph $\mathcal{G}$, or, equivalently, the vector of the row-sums of the matrix T, i.e. $d = Tu$. We furthermore fix two threshold values δ_i, $i = 1, 2$.

We first identify the quasi-symmetric high-valued elements in the matrix T by constructing an auxiliary matrix C with elements:

$$c_{ij} = \begin{cases} t_{ij}, & \text{if } t_{ij} \geq \delta_1 \\ 0 & \text{otherwise} \end{cases}. \tag{4}$$

It corresponds to the weighted adjacency matrix of the (undirected) sub-graph of $\mathcal{G}$ with edges connecting potential colluded agents.

We then construct the vector $\widetilde{d} = d - Cu$ and classify an agent i, for each $i = 1, \ldots, n$, as malicious if:

$$\widetilde{d}_i \leq \delta_2.$$

A proper selection of the threshold values δ_1 and δ_2 is crucial for achieving an accurate detection of the malicious agents.

The parameter δ_1 must be chosen sufficiently large to identify all possible candidates (agents which assign high trust scores to each other). So, in order not to exclude any of them, we fix it as:

$$\delta_1 = \min_{1 \leq j \leq n} \max_{1 \leq i \leq n} t_{ij}.$$

As to the threshold δ_2, in order to exclude from the first selection honest agents which receive anyway high trust values from the rest of the honest agents, we set it as:

$$\delta_2 = \frac{\sum_{i,j \in \mathcal{F}} (t_{ij} - c_{ij})}{|\mathcal{F}|}$$

where $\mathcal{F}$ is the index set of the potential colluded agents identified after the thresholding step (4) and c_{ij} are the elements of the matrix C. Such information can then be used to set the proper initial reputation vector in the Eigentrust model. Indeed, let $\mathcal{M}$ be the index set of the malicious users identified as above. The teleportation vector v has then elements v_i which are set to 0 if $i \in \mathcal{M}$, and to $1/(n - |\mathcal{M}|)$ otherwise.

It has to be noted that, in such a way, we assign the same amount of trust to the non-malicious agents, irrespective of their initial actual trust values.

3.3. *Our algorithm*

The strategy of Eigentrust is to penalize all the agents that are not pre-trusted (i.e., all the potential colluding agents), with the side-effect of computing an incorrect final reputation for those agents that are not in the pre-trusted set but also are not colluded. With the aim to avoid the above strong penalization of all the agents that are not pre-trusted, we propose an alternative approach to Eigentrust. The idea is to pre-process the set of the agents to determine a sub-set of suitable candidates to be considered as colluded, and then assign a low value of trustworthiness only to these "suspects", rather than to all the agents that are not in the pre-trusted set.

In particular, once the malicious are identified, we modify the matrix T into a new matrix $\widehat{T}$ as follows:

- Fix a value $\epsilon > 0$;
- Let $\widehat{t}_{ij} = \begin{cases} \epsilon, & i,j \in \mathcal{M} \\ t_{ij}, & \text{otherwise} \end{cases}$
- Make the matrix $\widehat{T}$ column stochastic by normalizing to 1 each column.

In the choice of ϵ the size of the matrix has to be taken into account, so to guarantee that it keeps a low value as n increases. This is achieved, for example, by setting $\epsilon = \beta/n$, with $\beta \leq 1$.

In our experiments, for computational purpose, we adopt the choice $\beta = 2 \cdot 10^{-3}$.

For the sake of completeness, the scheme describing the entire procedure (including the steps for the detection of malicious) is illustrated in Algorithm 1.

Numerical and experimental examples, as shown later, confirm that, in addition to detect colluding agents, this strategy allows to keep the reputations of all the honest agents unchanged.

4. An example

We propose an example to illustrate our approach and to compare it to the Eigentrust approach in the particular situation of a social network of 14 agents. We consider the trust matrix T given in Table 1. For the choice of α, we follow the traditional Eigentrust approach, where the value is fixed to 0.85.

Algorithm 1 Algorithm for malicious detection and reputation computation

Require: T, β, α
$v = [0, 0, \ldots, 0]^T$
$d \leftarrow T[1, 1, \ldots, 1]^T$
compute δ_1 as $\min\{\max\{t_{ij}, i = 1, \ldots, n\}, j = 1, \ldots, n\}$
$\mathcal{F} = \emptyset$
for $i = 1$ **to** n **do**
for $j = 1$ **to** n **do**
if $t_{ij} \geq \delta_1$ **then**
$c_{ij} \leftarrow t_{ij}$
$\mathcal{F} \leftarrow \mathcal{F} \cup \{i\}$ (i added only once)
else
$c_{ij} \leftarrow 0$
end if
end for
end for
$d \leftarrow d - C[1, 1, \ldots, 1]^T$
compute δ_2 as $\sum_{i,j} |t_{ij} - c_{ij}|/|\mathcal{F}|$
for $i = 1$ **to** n **do**
if $d_i > \delta_2$ **then**
$v_i = 1$
$\mathcal{M} \leftarrow \mathcal{M} \cup \{i\}$
end if
end for
$v \leftarrow v/\sum_i v_i$
for $i, j = 1$ **to** n **do**
if $i, j \in \mathcal{M}$ **then**
$t_{ij} \leftarrow \beta/n$
end if
end for
normalize T
solve $(\alpha T + (1 - \alpha)v[1, 1, \ldots, 1]^T)r = r$
return r

The suspected malicious users are the ones corresponding to the agents 8, 9 and 10. Moreover, the

Table 1 Matrices used for our example in Section 4

$$T = \begin{pmatrix}
0 & 0.4 & 0.39 & 0.33 & 0.26 & 0.23 & 0.2 & 0.001 & 0.002 & 0.003 & 0.25 & 0.19 & 0.23 & 0.24 \\
0.36 & 0 & 0.43 & 0.28 & 0.3 & 0.189 & 0.22 & 0.002 & 0.003 & 0.002 & 0.22 & 0.2 & 0.25 & 0.223 \\
0.42 & 0.41 & 0 & 0.19 & 0.28 & 0.195 & 0.23 & 0.001 & 0.005 & 0.001 & 0.23 & 0.21 & 0.196 & 0.196 \\
0.08 & 0.0351 & 0.07 & 0 & 0.1 & 0.14 & 0.11 & 0.003 & 0.001 & 0.009 & 0.1 & 0.095 & 0.12 & 0.09 \\
0.04 & 0.07 & 0.06 & 0.1 & 0 & 0.09 & 0.091 & 0.006 & 0.003 & 0.008 & 0.09 & 0.08 & 0.093 & 0.086 \\
0.02 & 0.019 & 0.01 & 0.06 & 0.02 & 0 & 0.1 & 0.009 & 0.01 & 0.006 & 0.08 & 0.1 & 0.055 & 0.076 \\
0.025 & 0.015 & 0.003 & 0.02 & 0.03 & 0.125 & 0 & 0.019 & 0.03 & 0.006 & 0.01 & 0.11 & 0.035 & 0.066 \\
0.01 & 0.0049 & 0.006 & 0.002 & 0.003 & 0.006 & 0.009 & 0 & 0.46 & 0.46 & 0.002 & 0.002 & 0.006 & 0.009 \\
0.009 & 0.006 & 0.003 & 0.006 & 0.001 & 0.002 & 0.001 & 0.42 & 0 & 0.42 & 0.003 & 0.001 & 0.005 & 0.003 \\
0.015 & 0.019 & 0.008 & 0.001 & 0.002 & 0.009 & 0.002 & 0.48 & 0.43 & 0 & 0.008 & 0.002 & 0.003 & 0.005 \\
0.009 & 0.009 & 0.005 & 0.003 & 0.001 & 0.008 & 0.006 & 0.01 & 0.009 & 0.001 & 0 & 0.002 & 0.002 & 0.002 \\
0.002 & 0.003 & 0.005 & 0.005 & 0.001 & 0.001 & 0.009 & 0.02 & 0.01 & 0.06 & 0.003 & 0 & 0.003 & 0.001 \\
0.004 & 0.008 & 0.001 & 0.002 & 0.001 & 0.002 & 0.01 & 0.009 & 0.02 & 0.02 & 0.003 & 0.004 & 0 & 0.003 \\
0.006 & 0.001 & 0.009 & 0.001 & 0.001 & 0.003 & 0.012 & 0.02 & 0.017 & 0.004 & 0.001 & 0.004 & 0.002 & 0
\end{pmatrix}$$

$$\widetilde{T} = \begin{pmatrix}
0.1925 & 0.2725 & 0.2705 & 0.2585 & 0.2445 & 0.2385 & 0.2325 & 0.1927 & 0.1929 & 0.1931 & 0.2425 & 0.2305 & 0.2385 & 0.2405 \\
0.2612 & 0.1892 & 0.2752 & 0.2452 & 0.2492 & 0.2270 & 0.2332 & 0.1896 & 0.1898 & 0.1896 & 0.2332 & 0.2292 & 0.2392 & 0.2338 \\
0.2650 & 0.2630 & 0.1810 & 0.2190 & 0.2370 & 0.2200 & 0.2270 & 0.1812 & 0.1820 & 0.1812 & 0.2270 & 0.2230 & 0.2202 & 0.2202 \\
0.0833 & 0.0743 & 0.0813 & 0.0673 & 0.0873 & 0.0953 & 0.0893 & 0.0679 & 0.0675 & 0.0691 & 0.0873 & 0.0863 & 0.0913 & 0.0853 \\
0.0657 & 0.0717 & 0.0697 & 0.0777 & 0.0577 & 0.0757 & 0.0759 & 0.0589 & 0.0583 & 0.0593 & 0.0757 & 0.0737 & 0.0763 & 0.0749 \\
0.0439 & 0.0437 & 0.0419 & 0.0519 & 0.0439 & 0.0399 & 0.0599 & 0.0417 & 0.0419 & 0.0411 & 0.0559 & 0.0599 & 0.0509 & 0.0551 \\
0.0399 & 0.0379 & 0.0355 & 0.0389 & 0.0409 & 0.0599 & 0.0349 & 0.0387 & 0.0409 & 0.0361 & 0.0369 & 0.0569 & 0.0419 & 0.0481 \\
0.0062 & 0.0052 & 0.0054 & 0.0046 & 0.0048 & 0.0054 & 0.0060 & 0.0042 & 0.0962 & 0.0962 & 0.0046 & 0.0046 & 0.0054 & 0.0060 \\
0.0046 & 0.0040 & 0.0034 & 0.0040 & 0.0030 & 0.0032 & 0.0030 & 0.0868 & 0.0028 & 0.0868 & 0.0034 & 0.0030 & 0.0038 & 0.0034 \\
0.0082 & 0.0090 & 0.0068 & 0.0054 & 0.0056 & 0.0070 & 0.0056 & 0.1012 & 0.0912 & 0.0052 & 0.0068 & 0.0056 & 0.0058 & 0.0062 \\
0.0065 & 0.0065 & 0.0057 & 0.0053 & 0.0049 & 0.0063 & 0.0059 & 0.0067 & 0.0065 & 0.0049 & 0.0047 & 0.0051 & 0.0051 & 0.0051 \\
0.0091 & 0.0093 & 0.0097 & 0.0097 & 0.0089 & 0.0089 & 0.0105 & 0.0127 & 0.0107 & 0.0207 & 0.0093 & 0.0087 & 0.0093 & 0.0089 \\
0.0069 & 0.0077 & 0.0063 & 0.0065 & 0.0063 & 0.0065 & 0.0081 & 0.0079 & 0.0101 & 0.0101 & 0.0067 & 0.0069 & 0.0061 & 0.0067 \\
0.0069 & 0.0059 & 0.0075 & 0.0059 & 0.0059 & 0.0063 & 0.0081 & 0.0097 & 0.0091 & 0.0065 & 0.0059 & 0.0065 & 0.0061 & 0.0057
\end{pmatrix}$$

$$\widehat{T} = \begin{pmatrix}
0 & 0.4000 & 0.3900 & 0.3300 & 0.2600 & 0.2300 & 0.2000 & 0.0100 & 0.0182 & 0.0250 & 0.2500 & 0.1900 & 0.2300 & 0.2400 \\
0.3600 & 0 & 0.4300 & 0.2800 & 0.3000 & 0.1890 & 0.2200 & 0.0200 & 0.0273 & 0.0167 & 0.2200 & 0.2000 & 0.2500 & 0.2230 \\
0.4200 & 0.4100 & 0 & 0.1900 & 0.2800 & 0.1950 & 0.2300 & 0.0100 & 0.0454 & 0.0083 & 0.2300 & 0.2100 & 0.1960 & 0.1960 \\
0.0800 & 0.0351 & 0.0700 & 0 & 0.1000 & 0.1400 & 0.1100 & 0.0300 & 0.0091 & 0.0750 & 0.1000 & 0.0950 & 0.1200 & 0.0900 \\
0.0400 & 0.0700 & 0.0600 & 0.1000 & 0 & 0.0900 & 0.0910 & 0.0600 & 0.0273 & 0.0667 & 0.0900 & 0.0800 & 0.0930 & 0.0860 \\
0.0200 & 0.0190 & 0.0100 & 0.0600 & 0.0200 & 0 & 0.1000 & 0.0900 & 0.0909 & 0.0500 & 0.0800 & 0.1000 & 0.0550 & 0.0760 \\
0.0250 & 0.0150 & 0.0030 & 0.0200 & 0.0300 & 0.1250 & 0 & 0.1900 & 0.2727 & 0.0500 & 0.0100 & 0.1100 & 0.0350 & 0.0660 \\
0.0100 & 0.0049 & 0.0060 & 0.0020 & 0.0030 & 0.0060 & 0.0090 & 0 & 0.0001 & 0.0001 & 0.0020 & 0.0020 & 0.0060 & 0.0090 \\
0.0090 & 0.0060 & 0.0030 & 0.0060 & 0.0010 & 0.0020 & 0.0010 & 0.0001 & 0 & 0.0001 & 0.0030 & 0.0010 & 0.0050 & 0.0030 \\
0.0150 & 0.0190 & 0.0080 & 0.0010 & 0.0020 & 0.0090 & 0.0020 & 0.0001 & 0.0001 & 0 & 0.0080 & 0.0020 & 0.0030 & 0.0050 \\
0.0090 & 0.0090 & 0.0050 & 0.0030 & 0.0010 & 0.0080 & 0.0060 & 0.1000 & 0.0818 & 0.0083 & 0 & 0.0020 & 0.0020 & 0.0020 \\
0.0020 & 0.0030 & 0.0050 & 0.0050 & 0.0010 & 0.0010 & 0.0090 & 0.2000 & 0.0909 & 0.4999 & 0.0030 & 0 & 0.0030 & 0.0010 \\
0.0040 & 0.0080 & 0.0010 & 0.0020 & 0.0010 & 0.0020 & 0.0100 & 0.0900 & 0.1818 & 0.1666 & 0.0030 & 0.0040 & 0 & 0.0030 \\
0.0060 & 0.0010 & 0.0090 & 0.0010 & 0.0010 & 0.0030 & 0.0120 & 0.2000 & 0.1545 & 0.0333 & 0.0010 & 0.0040 & 0.0020 & 0
\end{pmatrix}$$

members 1, 2 and 3 are good agents and receive high values of trust from all honest agents, whereas the remaining honest members receive only low trust values from all other agents.

We assume that $t_{ii} = 0$, $i = 1, \ldots, n$, so that the trust self assigned by a member is not considered.

In our example we fix $\delta_1 = 0.21$ and $\delta_2 = 0.96$.

Applying our threshold strategy, the agents 8, 9 and 10 are actually considered as the only malicious agents, so that $\mathcal{M} = \{8, 9, 10\}$.

Following the Eigentrust model, the teleportation vector is

$$v = \frac{1}{11}(1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1)^T$$

and the trust matrix $\widetilde{T}$, that is used in (3), is given in Table 1.

To construct the matrix $\widehat{T}$ following our new method, we fix $\epsilon = 0.00014$, taking into account the size of matrix T. For $i, j \in \mathcal{M} = \{8, 9, 10\}$, let $\widehat{t}_{ij} = \epsilon$ and by normalizing to 1 each column we get the column stochastic matrix $\widehat{T}$ in Table 1.

In Fig. 1 we illustrate, for comparison, the original trust matrix T and the $\widetilde{T}$, $\widehat{T}$ matrices obtained with the two different methods, where trust values are differentiated by color tone. It is evident that in the original trust matrix the rows 8, 9 and 10 exhibit unusual high values of trust and are classified as malicious users. We highlight that, unlike the Eigentrust matrix, the trust values of the honest agents remain unchanged in the matrix obtained by applying our

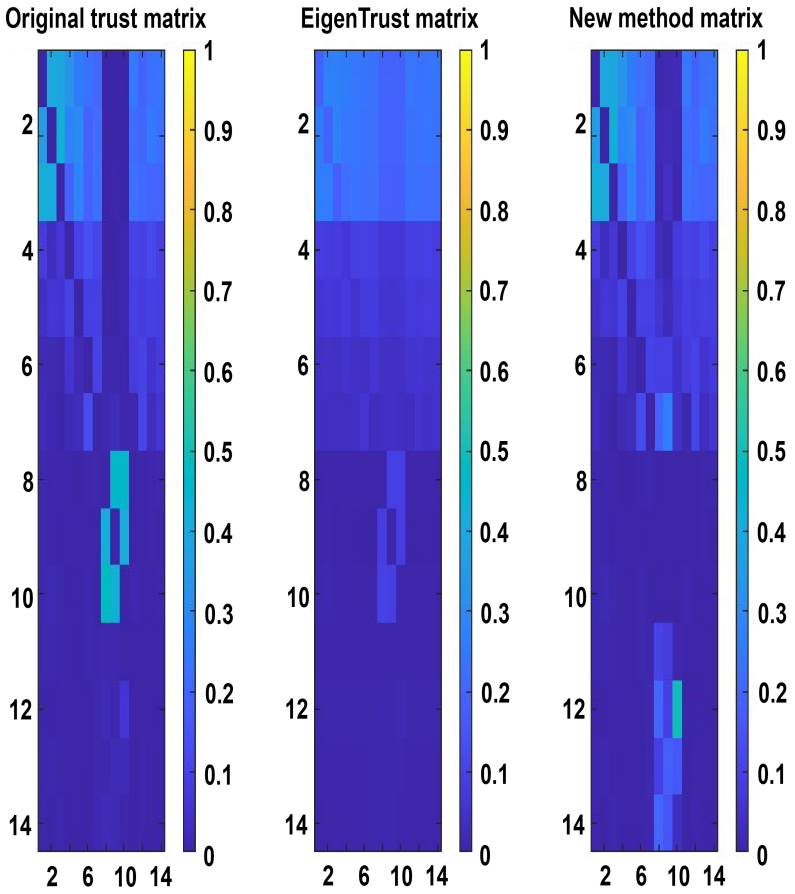

Figure 1. T, $\widetilde{T}$, $\widehat{T}$ matrices corresponding to the Example in Section 4.

method.

The respective corresponding reputation vectors, obtained by solving the eigensystem (2) with $\widetilde{T}$, $\widehat{T}$ matrices are plotted in Fig. 2. We note that the method we propose does not modify the real reputation scores of the honest agents unlike Eigentrust.

5. Error measure

To compare the results obtained by using different methods, we propose an error metric to measure how each method affects the reputation of the non colluded (pre-trusted) agents (once they are identified with our strategy) while lowering the reputation of malicious ones.

The idea is to evaluate the distance between the reputation values associated only with the pre-trusted users obtained with each algorithm and the solution to (2) once in the original matrix T the trust values associated with malicious agents are deleted.

More formally, let $\mathcal{P}$ be the set of indexes associated to the pre-trusted users, obtained as $\mathcal{P} = \{1, 2, \ldots, n\} \setminus \mathcal{M}$ and let $p = |\mathcal{P}|$. Let E be the $n \times p$ matrix whose columns are the standard unit coordinate (column) vectors $e^j = [0, \ldots, 1, \ldots, 0]^T$, $j \in \mathcal{P}$, with the only nonzero coefficient 1 at the j-th entry. Premultiplication by the transpose of such a matrix allows to extract only the rows associated to the pre-trusted-users, while postmoltiplication by E perfoms a similar extraction on the columns.

Suppose we have found a solution to (2), where T is replaced either by the matrix $\widetilde{T}$ as in (3) in the case of the Eigentrust method, with teleportation vector $v = \dfrac{1}{n}u$, or the matrix $\widehat{T}$ as in our strategy.

Once the reputation vector r has been computed

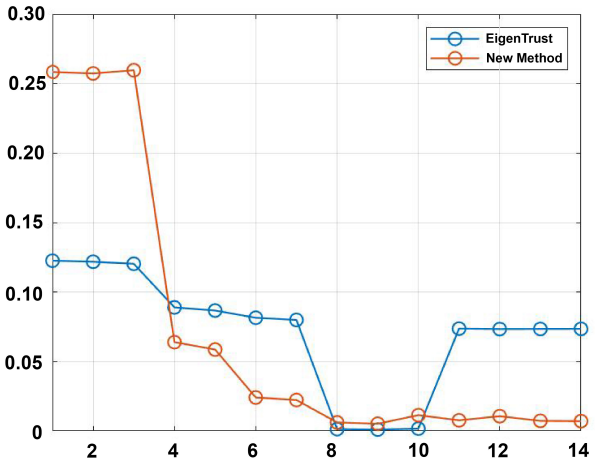

Figure 2. Reputation vectors obtained with the two different approaches.

Table 2 Errors with Eigentrust and our method, as functions of the number of agents and the percentage of malicious users

n	% malicious	Eigentrust		Our method	
		e_2	e_∞	e_2	e_∞
50	10%	$6.2 \cdot 10^{-1}$	$1.3 \cdot 10^{-2}$	$1.2 \cdot 10^{-3}$	$2.3 \cdot 10^{-3}$
	20%	$7.2 \cdot 10^{-2}$	$1.4 \cdot 10^{-1}$	$2.1 \cdot 10^{-3}$	$4.0 \cdot 10^{-3}$
	30%	$8.1 \cdot 10^{-2}$	$1.5 \cdot 10^{-1}$	$3.2 \cdot 10^{-3}$	$7.1 \cdot 10^{-3}$
	40%	$8.1 \cdot 10^{-2}$	$1.5 \cdot 10^{-1}$	$4.8 \cdot 10^{-3}$	$8.8 \cdot 10^{-3}$
	50%	$8.9 \cdot 10^{-2}$	$1.6 \cdot 10^{-1}$	$6.9 \cdot 10^{-3}$	$1.2 \cdot 10^{-2}$
100	10%	$4.8 \cdot 10^{-2}$	$1.1 \cdot 10^{-1}$	$4.2 \cdot 10^{-4}$	$9.5 \cdot 10^{-4}$
	20%	$4.9 \cdot 10^{-2}$	$1.1 \cdot 10^{-1}$	$6.7 \cdot 10^{-4}$	$1.5 \cdot 10^{-3}$
	30%	$5.5 \cdot 10^{-2}$	$1.2 \cdot 10^{-1}$	$1.0 \cdot 10^{-3}$	$2.2 \cdot 10^{-3}$
	40%	$5.9 \cdot 10^{-2}$	$1.2 \cdot 10^{-1}$	$1.6 \cdot 10^{-3}$	$3.2 \cdot 10^{-3}$
	50%	$6.4 \cdot 10^{-2}$	$1.3 \cdot 10^{-1}$	$2.3 \cdot 10^{-3}$	$4.2 \cdot 10^{-2}$
200	10%	$3.3 \cdot 10^{-2}$	$9.1 \cdot 10^{-2}$	$1.5 \cdot 10^{-4}$	$3.8 \cdot 10^{-4}$
	20%	$3.4 \cdot 10^{-2}$	$9.5 \cdot 10^{-2}$	$2.6 \cdot 10^{-4}$	$7.9 \cdot 10^{-4}$
	30%	$3.8 \cdot 10^{-2}$	$9.6 \cdot 10^{-2}$	$3.8 \cdot 10^{-4}$	$1.1 \cdot 10^{-3}$
	40%	$4.2 \cdot 10^{-2}$	$1.0 \cdot 10^{-1}$	$5.5 \cdot 10^{-4}$	$1.3 \cdot 10^{-3}$
	50%	$4.5 \cdot 10^{-2}$	$1.1 \cdot 10^{-1}$	$8.0 \cdot 10^{-4}$	$1.8 \cdot 10^{-3}$
300	10%	$2.7 \cdot 10^{-2}$	$7.5 \cdot 10^{-2}$	$8.4 \cdot 10^{-5}$	$2.5 \cdot 10^{-4}$
	20%	$2.9 \cdot 10^{-2}$	$7.9 \cdot 10^{-2}$	$1.4 \cdot 10^{-4}$	$3.6 \cdot 10^{-4}$
	30%	$3.0 \cdot 10^{-2}$	$8.6 \cdot 10^{-2}$	$1.9 \cdot 10^{-4}$	$5.5 \cdot 10^{-4}$
	40%	$3.5 \cdot 10^{-2}$	$9.9 \cdot 10^{-2}$	$3.0 \cdot 10^{-4}$	$8.7 \cdot 10^{-4}$
	50%	$3.8 \cdot 10^{-2}$	$1.0 \cdot 10^{-1}$	$4.2 \cdot 10^{-4}$	$1.1 \cdot 10^{-3}$
400	10%	$2.4 \cdot 10^{-2}$	$6.8 \cdot 10^{-2}$	$5.3 \cdot 10^{-5}$	$1.6 \cdot 10^{-4}$
	20%	$2.5 \cdot 10^{-2}$	$7.0 \cdot 10^{-2}$	$9.0 \cdot 10^{-5}$	$2.5 \cdot 10^{-4}$
	30%	$2.6 \cdot 10^{-2}$	$7.2 \cdot 10^{-2}$	$1.2 \cdot 10^{-4}$	$3.7 \cdot 10^{-4}$
	40%	$3.0 \cdot 10^{-2}$	$8.5 \cdot 10^{-2}$	$2.0 \cdot 10^{-4}$	$5.2 \cdot 10^{-4}$
	50%	$3.3 \cdot 10^{-2}$	$8.9 \cdot 10^{-2}$	$2.8 \cdot 10^{-4}$	$7.2 \cdot 10^{-4}$
500	10%	$2.1 \cdot 10^{-2}$	$6.9 \cdot 10^{-2}$	$3.7 \cdot 10^{-5}$	$1.1 \cdot 10^{-4}$
	20%	$2.3 \cdot 10^{-2}$	$7.1 \cdot 10^{-2}$	$6.6 \cdot 10^{-5}$	$1.8 \cdot 10^{-4}$
	30%	$2.3 \cdot 10^{-2}$	$7.2 \cdot 10^{-2}$	$9.5 \cdot 10^{-5}$	$2.6 \cdot 10^{-4}$
	40%	$2.6 \cdot 10^{-2}$	$7.4 \cdot 10^{-2}$	$1.3 \cdot 10^{-4}$	$3.9 \cdot 10^{-4}$
	50%	$2.9 \cdot 10^{-2}$	$8.0 \cdot 10^{-2}$	$2.1 \cdot 10^{-4}$	$6.1 \cdot 10^{-4}$

n	% malicious	Eigentrust		Our method	
		e_2	e_∞	e_2	e_∞
1000	10%	$1.5 \cdot 10^{-2}$	$4.9 \cdot 10^{-2}$	$1.3 \cdot 10^{-5}$	$4.2 \cdot 10^{-5}$
	20%	$1.6 \cdot 10^{-2}$	$5.3 \cdot 10^{-2}$	$2.2 \cdot 10^{-5}$	$7.4 \cdot 10^{-5}$
	30%	$1.7 \cdot 10^{-2}$	$6.0 \cdot 10^{-2}$	$3.4 \cdot 10^{-5}$	$1.1 \cdot 10^{-4}$
	40%	$1.8 \cdot 10^{-2}$	$5.7 \cdot 10^{-2}$	$4.8 \cdot 10^{-5}$	$1.4 \cdot 10^{-4}$
	50%	$2.0 \cdot 10^{-2}$	$6.8 \cdot 10^{-2}$	$7.3 \cdot 10^{-5}$	$2.1 \cdot 10^{-4}$
2500	10%	$0.9 \cdot 10^{-2}$	$3.4 \cdot 10^{-2}$	$3.4 \cdot 10^{-6}$	$1.2 \cdot 10^{-5}$
	20%	$1.0 \cdot 10^{-2}$	$3.4 \cdot 10^{-2}$	$5.7 \cdot 10^{-6}$	$2.1 \cdot 10^{-5}$
	30%	$1.0 \cdot 10^{-2}$	$3.6 \cdot 10^{-2}$	$8.6 \cdot 10^{-6}$	$2.8 \cdot 10^{-5}$
	40%	$1.2 \cdot 10^{-2}$	$4.0 \cdot 10^{-2}$	$1.2 \cdot 10^{-5}$	$4.1 \cdot 10^{-5}$
	50%	$1.3 \cdot 10^{-2}$	$4.4 \cdot 10^{-2}$	$1.8 \cdot 10^{-5}$	$5.9 \cdot 10^{-5}$
5000	10%	$6.9 \cdot 10^{-3}$	$2.5 \cdot 10^{-2}$	$1.2 \cdot 10^{-6}$	$4.5 \cdot 10^{-6}$
	20%	$7.3 \cdot 10^{-3}$	$2.7 \cdot 10^{-2}$	$2.0 \cdot 10^{-6}$	$7.7 \cdot 10^{-6}$
	30%	$7.8 \cdot 10^{-3}$	$2.8 \cdot 10^{-2}$	$3.0 \cdot 10^{-6}$	$1.2 \cdot 10^{-5}$
	40%	$8.4 \cdot 10^{-3}$	$2.8 \cdot 10^{-2}$	$4.4 \cdot 10^{-6}$	$1.8 \cdot 10^{-5}$
	50%	$9.2 \cdot 10^{-3}$	$3.2 \cdot 10^{-2}$	$6.5 \cdot 10^{-6}$	$2.1 \cdot 10^{-5}$
7500	10%	$5.7 \cdot 10^{-3}$	$2.2 \cdot 10^{-2}$	$6.5 \cdot 10^{-7}$	$2.3 \cdot 10^{-6}$
	20%	$6.0 \cdot 10^{-3}$	$2.2 \cdot 10^{-2}$	$1.1 \cdot 10^{-6}$	$4.2 \cdot 10^{-6}$
	30%	$6.4 \cdot 10^{-3}$	$2.4 \cdot 10^{-2}$	$1.6 \cdot 10^{-6}$	$6.5 \cdot 10^{-6}$
	40%	$6.9 \cdot 10^{-3}$	$2.5 \cdot 10^{-2}$	$2.4 \cdot 10^{-6}$	$9.3 \cdot 10^{-6}$
	50%	$7.5 \cdot 10^{-3}$	$2.5 \cdot 10^{-2}$	$3.5 \cdot 10^{-6}$	$1.3 \cdot 10^{-5}$
10000	10%	$4.9 \cdot 10^{-3}$	$1.9 \cdot 10^{-2}$	$4.2 \cdot 10^{-7}$	$1.7 \cdot 10^{-6}$
	20%	$5.1 \cdot 10^{-3}$	$1.9 \cdot 10^{-2}$	$7.1 \cdot 10^{-7}$	$2.5 \cdot 10^{-6}$
	30%	$5.5 \cdot 10^{-3}$	$1.9 \cdot 10^{-2}$	$1.0 \cdot 10^{-6}$	$3.5 \cdot 10^{-6}$
	40%	$6.0 \cdot 10^{-3}$	$2.1 \cdot 10^{-2}$	$1.5 \cdot 10^{-6}$	$5.5 \cdot 10^{-6}$
	50%	$6.5 \cdot 10^{-3}$	$2.4 \cdot 10^{-2}$	$2.3 \cdot 10^{-6}$	$8.2 \cdot 10^{-6}$
25000	10%	$3.8 \cdot 10^{-3}$	$1.5 \cdot 10^{-2}$	$1.1 \cdot 10^{-7}$	$4.2 \cdot 10^{-7}$
	20%	$4.0 \cdot 10^{-3}$	$1.5 \cdot 10^{-2}$	$1.9 \cdot 10^{-7}$	$5.9 \cdot 10^{-7}$
	30%	$4.3 \cdot 10^{-3}$	$1.6 \cdot 10^{-2}$	$3.0 \cdot 10^{-7}$	$7.8 \cdot 10^{-7}$
	40%	$4.7 \cdot 10^{-3}$	$1.9 \cdot 10^{-2}$	$4.4 \cdot 10^{-7}$	$9.1 \cdot 10^{-7}$
	50%	$5.1 \cdot 10^{-3}$	$1.9 \cdot 10^{-2}$	$5.8 \cdot 10^{-7}$	$2.1 \cdot 10^{-6}$

by one of the two methods, we extract only the values associated to the pre-trusted users, that is we compute $\widehat{r} = E^T \widehat{r}$.

We now use the original trust matrix T and solve the system

$$E^T T E \widetilde{r} = \widetilde{r},$$

which is associated to the "ideal" situation, where only the pre-trusted agents are taken into account.

After normalizing the vectors $\widehat{r}$ and $\widetilde{r}$, we evaluate the error both as

$$e_2 = \frac{\|\widetilde{r} - \widehat{r}\|_2}{\|\widetilde{r}\|_2} = \sqrt{\frac{\sum_{j=1}^{n}(\widetilde{r}_j - \widehat{r}_j)^2}{\sum_{j=1}^{n}\widetilde{r}_j^2}} \tag{5}$$

and

$$e_\infty = \frac{\|\widetilde{r} - \widehat{r}\|_\infty}{\|\widetilde{r}\|_\infty} = \frac{\max\limits_{1\le j\le n}|\widetilde{r}_j - \widehat{r}_j|}{\max\limits_{1\le j\le n}\widetilde{r}_j} \tag{6}$$

In Section 6 we perform several mathematical simulations, illustrating the results obtained for different values of the matrix size and different ratios of malicious users in the network. We make a quantitative analysis of the results by means of the error measures (5) and (6).

6. Experimental Results

We present the results of an experimental campaign of simulations for testing our method, comparing it with the Eigentrust algorithm.

We stress that we obtain the information on the malicious users straight from the matrix T in the pre-processing phase in both cases. In fact, the information on the pre-trusted agents needed by Eigentrust relates to the agents that are not identified as malicious in such a phase. In other words, in order to make the comparison between the two approaches meaningful, we use the same information both in Eigentrust and in our algorithm, i.e. we inform Eigentrust that the agents which are not identified as malicious could be considered honest, and then pre-trustable.

Our numerical code has been implemented using MATLAB Release 2022a following the steps summarized in our Algorithm 1. Our numerical experiments were run on a 64-bit workstation with a Intel(R) Core(TM) i7-10875H CPU @ 2.30 GHz and 128 GB of RAM.

The major computational load in the entire procedure corresponds to the computation of the solution to the system of linear equations (line 30). Our code uses the MATLAB function `eig` to solve it by finding the eigenvector corresponding to the eigenvalue 1, with a cost corresponding to $O(n^3)$. The running time of the larger experiment (corresponding to $n = 25.000$) was equal to 1821 seconds.

Our algorithm, if implemented by a centralized architecture, generates running times that are reasonable for small networks (10.000-50.000 nodes). We only refer to a centralized architecture, where a recomputation of the reputation values is needed whenever some users are added or removed from the system. Generally, on large-scale communities centralized architectures[42] are more expensive than distributed ones in terms of costs for data storage, retrieval tasks, computational and communication overheads.[25] In contrast, distributed architectures[35,54,77] may perform better than centralized approaches in terms of computational costs, although they are more complex to implement and more vulnerable to malicious attacks. It is of course possible to implement the algorithm in a distributed architecture fashion, to improve efficiency.

We considered different cases, for different values of n, i.e. the dimension of the social network, and different ratios of malicious users in the network. For each case we have performed several simulations. We report in Table 2 the average error obtained by both algorithms.

As shown in the table, our method is more effective than Eigentrust for each value of the dimension and for each value of the ratio, presenting a lower error than the one produced by Eigentrust.

As n increases, both errors decrease, but with our method the error becomes very small: for example, for $n = 25000$, with 20% of malicious members, the 2-norm error order is 10^{-7} against 10^{-3} of the Eigentrust algorithm.

Finally, we may observe that, as the ratio of malicious agents increases, the error increases for both methods, but our algorithm still produces a very small error (for $n = 25000$, the 2-norm error order with 10% of malicious members is 10^{-7} and it increases with 50% of malicious users but with the same 10^{-7}) magnitude.

In Fig. 3, we compare the 2-norm average errors obtained with the two methods, varying the number of users from 1000 to 25000 with a fixed ratio of 10% of malicious agents and in Fig. 4 and 5 we compare the 2-norm average errors obtained with the two methods in a social network of 5000 and 25000 users, varying the ratios of malicious users from 10% to 50%. The graphics in the other cases are analogous. Let us note that the behavior of the ∞-norm error is analogous to the one of 2-norm error. The significant improvement of the precision in determining the real reputation of the agents (i.e., that reputation that would be obtained if all the colluded agents were re-

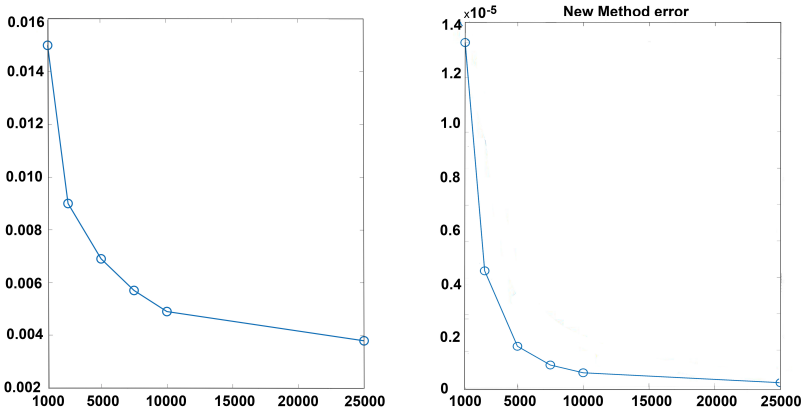

Figure 3. Average errors in 2-norm with 10% of malicious users for different values of the matrix size.

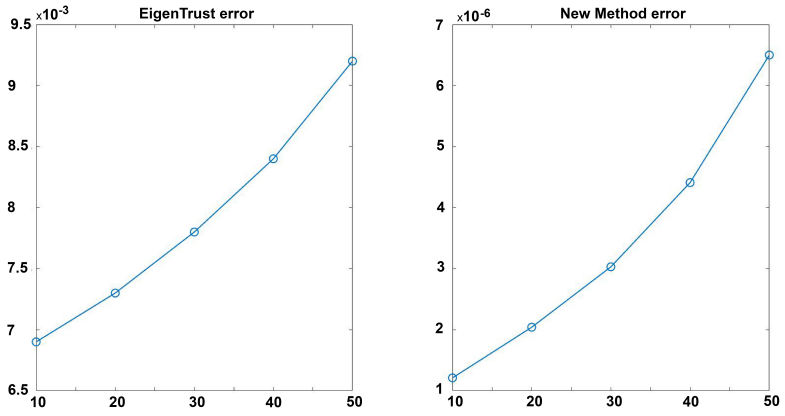

Figure 4. Average errors in 2-norm for a social network with $n = 5000$ and different ratios of malicious users

.

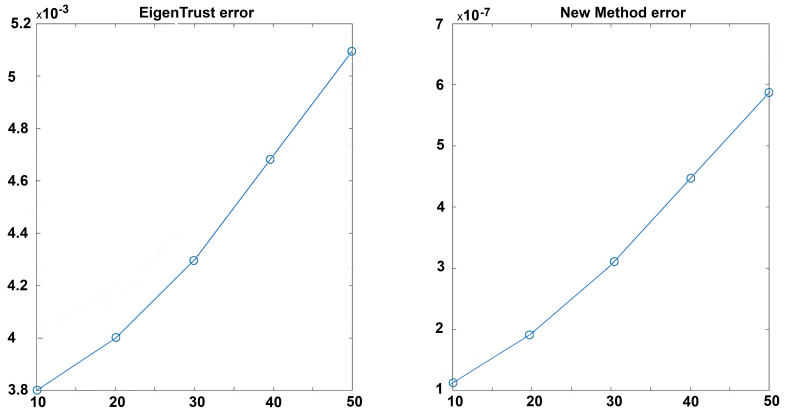

Figure 5. Average errors in 2-norm for a social network with $n = 25000$ and different ratios of malicious users

.

moved from the community) is due to the fact that our algorithm, differently from Eigentrust, avoids a strong penalization of pre-trusted (non-colluding) agents, while lowering the final reputation of malicious ones. This is possible since our algorithm performs a specific pre-processing phase (not present in EigentTrust) to detect potential colluded candidates. This then allows to assign a low value of trustworthiness only to these colluded candidates, without penalizing all the non pre-trusted agents (which is the strategy adopted by EigentTrust and that leads to the side-effect of generating incorrect reputation values for these non pre-trusted agents).

7. Conclusion

In this paper, we have focused on the problem of identifying, in a social network, the trustworthiness of an agent (human or software entity), in order to detect malicious actors and marginalize or expel them from the community.

In this context, several TRR systems have been proposed in the literature, and here we have examined the particular case represented by the well-known algorithm Eigentrust. This algorithm is recognized as one of the most effective solution both to measure the reputation in a set of social agents and as benchmark or training assistant. However, we see that Eigentrust, in order to detect malicious agents, uses some additional information about agents that can be a priori considered particularly trustworthy, rewarding them in terms of reputation, while the other agents are penalized.

In this setting, we have highlighted a possible limitation in the Eingentrust algorithm, observing that it increases the reputation of all the pre-trusted agents, regardless of their reliability. This way, the method is capable to effectively recognize colluding agents, but producing the side effect to flattening the differences, in terms of reliability, between honest agents.

To address the problem above, in this paper we have proposed a different strategy, based on a decrement of the fraudulent trust values that colluding agents mutually exchange. Our strategy introduces the advantage, with respect to Eigentrust, of estimating the reputation values of the honest actors in a manner more adherent to the actual reliability of these agents. This elevated precision of our method is particularly important, when the reputation of the agents is computed in a distributed environment, when different reputation values are continuously combined from different sources.

In order to measure this improvement of effectiveness we have introduced a metric of error to quantitatively determine how much an algorithm for the identification of malicious agents modifies the correct reputation values of the honest agents.

We have used such a metric in an experimental simulation, in which we have compared the effectiveness of our result with that generated by Eigentrust. We have shown that our method is more effective than Eigentrust in determining reputation values, presenting an error, which is about a thousand times lower than that produced by Eingentrust on medium-size social networks.

7.1. *Limitations and Future Work*

In this paper, we have considered only one category of malicious agents, precisely colluding agents. We have designed a very simple strategy to determine malicious candidates in a pre-processing phase, before computing the reputation values of the agents. The same idea could be theoretically applied to other types of malicious agents, but this requires to design apposite strategies which is not a trivial task. Our ongoing research is currently devoted to facing such an issue. We thus highlight that our approach is limited to environments presenting only the collusion as malicious activities. It could be applicable to more diverse and realistic settings only after having designed suitable strategies to effectively detect other types of malicious agents in the aforementioned pre-processing phase. Moreover, we notice that the results of our algorithm in terms of effectiveness strictly depends on the richness of the information contained in the network, that is maximum for a complete network. Conversely, for other types of less rich and more sparse network it would inevitably lead to a deterioration of performances. Finally, our algorithm is currently limited to be applied on static networks. The impact of possible dynamics of the network, very useful in realistic situations, is a crucial issue to explore.

A future work is currently devoted to design applications of our algorithm to real, dynamic, and large social networks, which will allow to deal with the above limitations. In large networks, another

study is addressed to quickly recognize reputation anomalies by combining our method, graphical representation of network properties[17,26] and NNs. To this aim, there exist some NN models potentially interesting (i.e., recurrent NNs,[57,60] competitive NNs,[78,84] deep learning and graph NNs,[29,50,70] and reinforced learning based NNs[13,81]). Also distributed[8] and federated[83] NNs are promising areas. In the former, Eigentrust can be used to assess trust among nodes participating in the training/execution of a NN to give more weight to the most trusted nodes. Similarly in federated NN systems, Eigentrust can assign greater weight to the contributions of the most reliable participants and mitigate the influence of potentially malicious ones. In these two cases, integrating our version of Eigentrust with NNs enables improved trust management and security, providing better performance to NNs.

Acknowledgements

This work was supported by the Project CAL.HUB.RIA funded by the Italian Ministry of Health,Project CUP: F63C22000530001. Local Project CUP: C33C22000540001.

Bibliography

1. Z. Abrams, R. Mcgrew, S. Plotkin, A non-manipulable trust system based on eigentrust, ACM SIGecom Exchanges 5 (4), 21–30, 2005.
2. A. Adamopoulou, A. Symeonidis, A simulation testbed for analyzing trust and reputation mechanisms in unreliable online markets, Electronic Commerce Research and Applic. 13 (5), 368–386, 2014.
3. M. Aghababaei, M. Koliou, Community resilience assessment via agent-based modeling approach, Computer-Aided Civil and Infrastructure Engineering, 38(7):920–939, 2023.
4. A. Alexiadis, A. Veliskaki, A. Nizamis, et al., A smarthome conversational agent performing implicit demand-response application planning, Integrated Computer-Aided Engineering, 29(1):43–61, 2022.
5. A. Alhussain, H. Kurdi, L. Altoaimy, A neural network-based trust management system for edge devices in peer-to-peer networks, Computers, Materials & Continua, 59(3), 805–815, 2019.
6. A. Alhussain, H. Kurdi, L. Altoaimy, Managing trust and detecting malicious groups in peer-to-peer iot Networks, Sensors, 21(13):4484, 2021.
7. A. Ahmed, K. Bakar, M. Channa, K. Haseeb, A. Khan, A survey on trust based detection and isolation of malicious nodes in ad-hoc and sensor networks, Frontiers of Computer Science 9 (2), 280–296, 2015.
8. R. Anil, G. Pereyra, A. Passos, et al., Large scale distributed neural network training through online distillation, preprint arXiv:1804.03235, 2018.
9. A. Bidgoly, B. Ladani, Benchmarking reputation systems: A quantitative verification approach, Computers in Human Behavior 57, 274–291, 2016.
10. A. Bidgoly, B. Ladani, Modelling and quantitative verification of reputation systems against malicious attackers, The Computer Journal 58 (10), 2567–2582, 2015
11. S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, Computer networks and ISDN systems, 30(1-7):107–117, 1998.
12. L. Cabral, A. Hortacsu, The dynamics of seller reputation: Evidence from ebay, Journal of Industrial Economics 58 (1), 54–78, 2010.
13. B. Chen, S. Yang, C. Kuo, J. Chen, et al., Neuro-inspired reinforcement learning to improve trajectory prediction in reward-guided behavior, International Journal of Neural Systems, 32(09):2250038, 2022.
14. M. Cheng, S. Nazarian, P. Bogdan, There is hope after all: Quantifying opinion and trustworthiness in neural networks, Frontiers in artificial intelligence, 3:54, 2020.
15. M. Cotronei, S. Giuffré, A. Marcianó, D. Rosaci, G. M. L. Sarné, Identifying Colluding Actors in Social Communities by Reputation Measures, in: The 2nd Int. Conf. on Applied Intelligence and Informatics (AII 2022), Springer Nature, 347–359, 2023.
16. P. De Meo, F. Messina, M. N. Postorino, D. Rosaci, G. M. L. Sarné, A reputation framework to share resources into iot-based environments, in: 14th Int. Conf. Networking, Sensing and Control, IEEE, 513–518, 2017.
17. D. Décary-Hétu, B. Dupont, The social network of hackers, Global Crime, 13(3):160–175, 2012.
18. A. Dorri, S. Kanhere, R. Jurdak, Multi-agent systems: A survey, IEEE Access 6, 28573–28593, 2018.
19. A. El Sheikh, M. Bevilacqua, R. Navigli, et al., Integrating personalized PageRank into neural word sense Disambiguation, Proc. of the 2021 Conf. on Empirical Methods in Natural Language Processing, 9092–9098. ACL, 2021.
20. A. Esmalian, W. Wang, A. Mostafavi, Multi-agent modeling of hazard–household–infrastructure nexus for equitable resilience assessment, Computer-Aided Civil and Infrastructure Engineering, 37(12):1491–1520, 2022.
21. X. Fan, L. Liu, M. Li, Z. Su, Eigentrustp++: Attack resilient trust management, in: 8th Int. Conf. on Collaborative Computing: Networking, Applications and Worksharing, IEEE, 416–425, 2012.
22. W. Fang, W. Zhang, W. Chen, T. Pan, et al., Trust-based attack and defense in wireless sensor networks: a survey, Wireless Communications and Mobile Computing, 1–20, 2020.
23. G. Fortino, F. Messina, D. Rosaci, G. M. L. Sarné, Using blockchain in a reputation-based model for

grouping agents in the internet of things, IEEE Trans. on Engineering Management 67 (4), 1231–1243, 2019.
24. G. Fortino, F. Messina, D. Rosaci, G. M. Sarnè, Resiot: An iot social framework resilient to malicious activities, IEEE/CAA Journal of Automatica Sinica 7 (5), 1263–1278, 2020.
25. G. Fortino, F. Messina, D. Rosaci, G. ML Sarné, C. Savaglio, A trust-based team formation framework for mobile intelligence in smart factories, IEEE Trans. on Industrial Informatics, 16(9):6133–6142, 2020.
26. L. Freeman, Visualizing social networks, Journal of social structure, 1(1):4, 2000.
27. K. Fullam, T. Klos, G. Muller, et al., A specification of the agent reputation and trust (art) testbed: experimentation and competition for trust in agent societies, in: 4th int. joint conf. on AAMAS, 512–518, 2005.
28. S. Gao, T. Yu, J. Zhu, W. Cai, T-pbft: An eigentrust-based practical byzantine fault tolerance consensus algorithm, China Communic. 16 (12), 111–123, 2019.
29. J. Gasteiger, A. Bojchevski, S. Günnemann, Predict then propagate: Graph neural networks meet personalized pagerank, preprint arXiv:1810.05997, 2018.
30. S. Ghasempouri, B. Ladani, Modeling trust and reputation systems in hostile environments, Future Generation Computer Systems 99, 571–592, 2019.
31. T. Grandison, M. Sloman, Trust management tools for internet applications, in: Int. Conf. on Trust Management, Springer, 91–107, 2003.
32. M. Gutierrez Soto, H. Adeli, Multi-agent replicator controller for sustainable vibration control of smart structures, Journal of Vibroengineering, 19(6):4300–4322, 2017.
33. S. Hayne, H. Wang, L. Wang, Modeling reputation as a time-series: Evaluating the risk of purchase decisions on ebay, Decision Sciences 46 (6), 1077–1107, 2015
34. S. Hendrikse, J. Treur, S. Koole, Modeling emerging interpersonal synchrony and its related adaptive short-term affiliation and long-term bonding: a second-order multi-adaptive neural agent model, International Journal of Neural Systems, 2350038, 2023.
35. F. Hendrikx, K. Bubendorfer, R. Chard, Reputation systems: A survey and taxonomy, Journal of Parallel and Distributed Computing, 75, 184–197, 2015.
36. K. Hoffman, D. Zage, C. Nita-Rotaru, A survey of attack and defense techniques for reputation systems, ACM Computing Surveys, 42 (1), 1–31, 2009.
37. http://www.ebay.com , 2022.
38. T. Huynh, N. Jennings, N. Shadbolt, An integrated trust and reputation model for open multi-agent systems, Autonomous Agents and Multi-Agent Systems 13 (2), 119–154, 2006.
39. R. Jansen, T. Kaminski, F. Korsakov, A. St Croix, D. Selifonov, A priori trust vulnerabilities in eigentrust, Tech. Rep. Univ. Minnesota, 1–11, 2008.
40. H. Jnanamurthy, S. Singh, Detection and filtering of collaborative malicious users in reputation system using quality repository approach, in: 2013 Conf. Adv. in Computing, Communications, Informatics, IEEE, 466–471, 2013.
41. A. Jøsang, J. Golbeck, Challenges for robust trust and reputation systems, in: Proc. of the 5th Int. Workshop on Security and Trust Management, Saint Malo, France, vol. 5, 1–12, 2009.
42. A. Jøsang, R. Ismail, C. Boyd, A Survey of Trust and Reputation Systems for Online Service Provision, Decision Support System, 43(2), 618–644, 2005.
43. S. Kamvar, M. Schlosser, H. Garcia-Molina, The eigentrust algorithm for reputation management in p2p networks, in: Proc. of the 12th int. conf. on World Wide Web, ACM, 640–651, 2003.
44. R. Kerr, R. Cohen, Treet: The trust and reputation experimentation and evaluation testbed, Electronic Commerce Research 10 (3), 271–290, 2010.
45. Vladik Kreinovich, Linear neural networks revisited: From pagerank to family happiness, How Uncertainty-Related Ideas Can Provide Theoretical Explanation For Empirical Dependencies, 83–92, 2021.
46. H. Kurdi, Honestpeer: An enhanced eigentrust algorithm for reputation management in p2p systems, J. of King Saud Univ.-Computer and Information Sciences 27 (3), 315–322, 2015.
47. Z Li,H Adeli, New adaptive robust h^{∞} control of smart structures using synchrosqueezed wavelet transform and recursive least-squares algorithm, Engineering Applications of Artificial Intelligence, 116:105473, 2022.
48. G. Liu, Q. Yang, H. Wang, A. Liu, Trust assessment in online social networks, IEEE Trans. on Dependable and Secure Computing 18 (2), 994–1007, 2019.
49. K. Lu, J. Wang, M. Li, An eigentrust dynamic evolutionary model in p2p file-sharing systems, Peer-to-Peer Networking and Applic. 9 (3), 599–612, 2016.
50. R. Maisano, G. Foresti A sentiment analysis anomaly detection system for cyber intelligence, International Journal of Neural Systems, 33(02):2350003, 2023.
51. S. V. Mallapur, S. R. Patil, Fuzzy logic based trusted candidate selection for stable multipath routing, IJ Information Tech. Computer Science 6, 12–21, 2015.
52. F. G. Mármol, G. M. Pérez, Security threats scenarios in trust and reputation models for distributed systems, computers & security 28 (7), 545–556, 2009.
53. F. G. Mármol, G. M. Pérez, Trmsim-wsn, trust and reputation models simulator for wireless sensor networks, in: 2009 Int. Conf. on Communications, IEEE, 1–5, 2009.
54. F. Messina, G. Pappalardo, D. Rosaci, C. Santoro, G. M. L. Sarné, A trust-aware, self-organizing system for large-scale federations of utility computing infrastructures, Future Generation Computer Systems, 56, 77–94, 2016.
55. F. Michel, J. Ferber, A. Drogoul, Multi-agent systems and simulation: A survey from the agent community's perspective, in: Multi-Agent Systems, CRC Press, pp. 17–66, 2018.

56. G Miebs, M Wójcik, A Karaszewski, et al., Predicting a time-dependent quantity using recursive generative query network, International Journal of Neural Systems, 32(11):2250056, 2022.
57. F. Moradi, H. Mohammadi, M. Rezaei, P. Sariaslani, N. Razazian, H. Khazaie, H. Adeli, A novel method for sleep-stage classification based on sonification of sleep electroencephalogram signals using wavelet transform and recurrent neural network, European Neurology, 83(5):468–486, 2020.
58. R. Mujumdar, S. Kumar, Hawkeye: A robust reputation system for community-based countermisinformation, In Proc. of the 2021 IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining, 188–192, 2021.
59. L. Page, S. Brin, R. Motwani, T. Andwinograd, The PageRank citation ranking: Bringing order to the web. Tech. rep. Stanford Digital Library Technologies Project, 1998.
60. C. Perez-Ramirez, J. Amezquita-Sanchez, M. Valtierra-Rodriguez, H. Adeli, et al., Recurrent neural network model with bayesian training and mutual information for response prediction of large buildings, Engineering Structures, 178:603–615, 2019.
61. P. Petruzzi, J. Pitt, D. Busquets, Electronic social capital for self-organising multi-agent systems, ACM Trans. on Autonomous and Adaptive Systems 12 (3), 1–25, 2017.
62. B. Pourghebleh, K. Wakil, N. Navimipour, A comprehensive study on the trust management techniques in the internet of things, IEEE Internet of Things Journal 6 (6), 9326–9337, 2019.
63. M. N. Postorino, G. ML Sarné, Agents meet traffic simulation, control and management: A review of selected recent contributions, Proc. 17th Work. "from Objects to Agents", WOA, vol. 1664, 112–117, 2016.
64. P. Resnick, R. Zeckhauser, Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system, in: The Economics of the Internet and E-commerce, Emerald Publishing, 127–157, 2002.
65. Y. Rizk, M. Awad, E. Tunstel, Decision making in multiagent systems: A survey, IEEE Trans. on Cognitive and Develop. Systems 10 (3), 514–529, 2018.
66. D Rosaci, G. ML Sarné, S. Garruzzo, Integrating trust measures in multiagent systems, International Journal of Intelligent Systems, 27(1):1–15, 2012.
67. S. Sajjad, S. Bouk, M. Yousaf, Neighbor node trust based intrusion detection system for wsn, Procedia Computer Science 63, 183–188, 2015.
68. A. Salehi-Abari, T. White, Dart: A distributed analysis of reputation and trust framework, Computational Intelligence 28 (4), 642–682, 2012.
69. F. Scarselli, S. Yong, M. Hagenbuchner, A. Tsoi, et al., Graph neural networks for ranking web pages, The 2005 IEEE/WIC/ACM Int. Conf. on Web Intelligence, 666–672, 2005.
70. M. Shahabi, A. Shalbaf, B. Nobakhsh, R. Rostami, R. Kazemi, Attention-based convolutional recurrent deep neural networks for the prediction of response to repetitive transcranial magnetic stimulation for major depressive disorder, International Journal of Neural Systems, 33(02):2350007, 2023.
71. A. Solomon, E. Olajubu, I. Ogundoyin, G. Aderounmu, A trust model for detecting device attacks in mobile ad hoc ambient home network, I.J. Advanced Pervasive Ubiquitous Computing 8(2), 16–37, 2016.
72. S. Vavilis, M. Petković, N. Zannone, A reference model for reputation systems, Decision Support Systems 61, 147–154, 2014.
73. S. Vitabile, V. Conti, C. Militello, F. Sorbello, An extended jade-s based framework for developing secure multi-agent systems, Computer Standards & Interfaces, 31(5):913–930, 2009.
74. Y.-F. Wang, Y. Hori, K. Sakurai, Characterizing economic and social properties of trust and reputation systems in p2p environment, Journal of Computer Science and Technology 23 (1), 129–140, 2008.
75. H Ying, H Zhou, A Degani, R Sacks, A two-stage recursive ray tracing algorithm to automatically identify external building objects in building information models, Computer-Aided Civil and Infrastructure Engineering, 37(8):991–1009, 2022.
76. Z. Xiao, Y. Deng, Graph embedding-based novel protein interaction prediction via higher-order graph convolutional network, PloS one, 15(9):e0238915, 2020.
77. L. Xiong, L. Liu, Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities, IEEE trans. on Knowledge and Data Engineering 16 (7), 843–857, 2004.
78. Y. Xue, Y. Zhang, F. Neri, A method based on evolutionary algorithms and channel attention mechanism to enhance cycle generative adversarial network performance for image translation, International Journal of Neural Systems, 33(05):2350026, 2023.
79. G. Zacharia, A. Moukas, P. Maes, Collaborative reputation mechanisms for electronic marketplaces, Decision support systems, 29(4):371–388, 2000.
80. Y. Zhang, D. Shi, Y. Wu, et al., Multi-agent feature learning and integration for mixed cooperative and competitive environment, in: The 32nd Int. Conf. on Tools with Artificial Intelligence, IEEE, 9–14, 2020.
81. G. Zhang, X. Zhang, H. Rong, et al., A layered spiking neural system for classification problems, International Journal of Neural Systems, 32(08):2250023, 2022.
82. R. Zhou, K. Hwang, Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing, IEEE Trans. on parallel and distributed systems, 18(4):460–473, 2007.
83. H. Zhu, H. Zhang, Y. Jin, From federated learning to federated neural architecture search: a survey, Complex & Intelligent Systems, 7:639–657, 2021.
84. H. Zhu, J. Wang, S. Wang, et al., An evolutionary attention-based network for medical image classification, International Journal of Neural Systems, 33(03):2350010, 2023.