Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

From EigenTrust To SHAPE-Trust

Bachelor's Thesis

Jan Rutterle

Study program: Open Informatics
Specialisation: Artificial Intelligence and Computer Science
Supervisor: doc. Ing. Tomáš Kroupa, Ph.D.

Prague, May 2024

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Rutterle Jan**	Personal ID number:	**507642**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Open Informatics**		
Specialisation:	**Artificial Intelligence and Computer Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

From EigenTrust to SHAPE-Trust

Bachelor’s thesis title in Czech:

Od EigenTrust k SHAPE-Trust

Guidelines:

The EigenTrust algorithm [1] is a reputation management system used in peer-to-peer networks to assess the trustworthiness of peers. It assigns each peer a trust score based on the history of interactions, with higher scores for peers that consistently provide reliable content. Trust scores are calculated using a decentralized, iterative process similar to Google's PageRank, where the trustworthiness of a peer is influenced by the trustworthiness of its neighbors. This system is dynamic, adjusting over time based on peer behavior, and includes measures to prevent abuse by malicious actors. The SHAPE-Trust (SHApley value for PEer-to-peer Trust) is a novel alternative to the EigenTrust algorithm, utilizing the Shapley value from game theory to compute trust scores based on local trust values among peers. This bachelor thesis aims to achieve several objectives:
1. Implement both the EigenTrust and SHAPE-Trust algorithms using the Julia programming language.
2. Conduct a comparative analysis of the trust scores generated by both methods. This comparison will utilize real or simulated transaction data from small to moderately sized peer-to-peer networks, focusing on how each method ranks peers.
3. Examine SHAPE-Trust's characteristics, either theoretically or through practical experiments, highlighting its distinctions from EigenTrust. This includes exploring situations where iterative methods for calculating eigenvalues might not converge [2] and examining cases where EigenTrust's transitive trust scoring is pivotal.
4. (Optional) Investigate potential applications of SHAPE-Trust outside of peer-to-peer network environments.

Bibliography / sources:

[1[Kamvar, S. D., Schlosser, M. T. & Garcia-Molina, H. The Eigentrust algorithm for reputation management in P2P networks. in Proceedings of the 12th international conference on World Wide Web 640–651 (Association for Computing Machinery, 2003).
[2] Afanador, J., Oren, N., Baptista, M. & Araujo, M. From Eigentrust to a Trust-measuring Algorithm in the Max-Plus Algebra. ECAI 2020.
[3] Bandhana, A., Kroupa, T., Garcia, S. Trust in Shapley: A Cooperative Quest for Global Trust in P2P Network. Přijato na konferenci AAMAS 2024.

Name and workplace of bachelor’s thesis supervisor:

doc. Ing. Tomáš Kroupa, Ph.D. Artificial Intelligence Center FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **23.01.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

doc. Ing. Tomáš Kroupa, Ph.D. Supervisor’s signature	prof. Dr. Ing. Jan Kybic Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Used Software

- ChatGPT (OpenAI) for text rephrasing and code tips for packages in Julia (https://chatgpt.com/)

Abstract

Peer-to-peer (P2P) networks have been rising in popularity, therefore the need to create and maintain a secure environment is pivotal. Security of a P2P network is preserved via Trust (Reputation) Management Systems (TMSs). Those systems use some form of local trust values, to compute global trust assigned to each peer in the network. This thesis focus on a comparison of three TMSs applied in simulated networks with specific properties. EigenTrust, is an iterative algorithm, which assigns a trust score to each peer through aggregating the opinions. MaxTrust is an expansion of EigenTrust to a non-linear algebra called Max-Plus algebra. SHAPE-Trust uses coalitional game theory and Shapley value, to fairly distribute the global trust scores. All algorithms were implemented in the Julia programming language. In case of EigenTrust only the centralized versions were implemented, because the remaining algorithms do not have a decentralized version yet.

Keywords: Trust (Reputation) Management Systems; P2P network; EigenTrust; Max-Trust; SHAPE-Trust.

Abstrakt

Peer-to-peer sítě jsou stále populárnější, a proto je zásadní v nich vytvořit a udržet bezpečné prostředí. Bezpečnost P2P sítě je zajištěna prostřednictvím systémů pro správu důvěry (reputace) (TMS). Tyto systémy používají nějakou formu lokálních hodnot důvěry k výpočtu globální důvěry přidělené každému uzlu v síti. Tato práce se zaměřuje na porovnání tří TMS aplikovaných ve simulovaných sítích se specifickými vlastnostmi. EigenTrust je iterativní algoritmus, který přiřazuje skóre důvěry každému uzlu prostřednictvím agregace názorů. MaxTrust je rozšíření EigenTrust do nelineární algebry zvané Max-Plus algebra. SHAPE-Trust používá koaliční teorii her a Shapleyovu hodnotu k spravedlivému rozdělení globálních hodnot důvěry. Všechny algoritmy byly implementovány v programovacím jazyce Julia. V případě EigenTrust byly implementovány pouze centralizované verze, protože zbylé algoritmy zatím nemají decentralizovanou verzi.

Klíčová slova: Systémy řízení důvěry (reputace), P2P síť, EigenTrust; MaxTrust; SHAPE-Trust.

Contents

List of Figures

List of Tables

Introduction

In our modern world the most known type of digital networks for file sharing is client/server architecture but it has some limitations in security. A good alternative to clients/server networks are peer-to-peer networks (P2P). In P2P every peer acts as a client and a server. This means that peers provide for example storage for data or processing power but at the same time receive data from other peers. The success of these kind of networks was proven by Napster or Gnutella [1]. Because there is no central authority (a single server) the possibility of a failure of the whole network because of a single node failure is minimal. Still there are security threats and possibilities of malicious attacks. Some threats may come from the presence of malicious peers, that tries to damage the network from inside. To help with recognizing these malicious peers from others, reputation management systems are introduced.

Reputation or Trust management systems (TMS) use trust to determine whom to block and who is reliable. Those systems mostly use some form of local trust values, which are based on the direct interactions of two peers, to calculate global trust. Global trust values are assigned to each peer in a network according to how much they are trusted in the network. These trust management models are essential for security measures. There is already series of models which can differ in their approach but all have a common purpose: establishing a framework to minimize risk of unwanted peer communication.

Each TMS is different with its approach and field of use. Therefore the local trust values can be very simple (calculated only based on good and bad transactions) or somewhat more complicated when there are more factors in each transaction that can be rated. There are many papers that describe the differences and applications of these various algorithms [2, 3], but the most cited one for its simplicity and effectivity is the EigenTrust.

Throughout this thesis the local trust values of a P2P network can be viewed as weighted directed graph (trust graph) $\mathcal{G} = (N, E)$, where $N = \{1, \dots, n\}$ is the set of peers. E is the set of edges (i, j), where the weights describe how much peer i trusts peer j. This graph can be written as a trust matrix $A \in \mathbb{R}^{n \times n}$, where every edge $(i, j) \in E$ can be found in this matrix as a_{ij}. However, if $a_{ij} = 0$, it can be either because the edge weigh is zero or the edge does not exist. This problem appears only in EigenTrust. In the rest of discussed algorithm, there are measures for differentiating between the two cases. It will be also assumed that $\mathcal{G}$ is simple, meaning there are not loops (i, i) (peers cannot assign trust to themselves). The trust matrix is essential part of the input for each presented algorithm, but its form may be a little different for each of them. For more detailed description of used graphs and matrices with some examples see Section 1.1.

This thesis will be focus on comparison of EigenTrust and its expansion to Max-Plus algebra with a new algorithm called SHAPE-Trust [4]. We will introduce their methodologies and theoretical background, cover their strong and weak sides and compare them in numerical experiments. The implementation of these algorithms can be found in the appendix.

Chapter 1

EigenTrust

EigenTrust calculates global trust values from normalized trust matrix of a network as an eigenvector of this matrix. For this it uses Power Method which guarantees to converge to the dominant eigenvector if some conditions are satisfied. In this part the assumptions of EigenTrust will be introduced as well as the description of the main idea, pseudocode of the algorithm and finally some issues that come along with those conditions. This thesis will be mainly focused on cases, where there are no pre-trusted peers and the trust matrices are reducible and therefore the EigenTrust should perform poorly.

1.1 Normalized Trust Matrix

The normalized trust matrix $C = [c_{ij}]$ is a square matrix of local trust values which every peer assigns to his companions. Each row represents normalized trust values from a peer to his companions, meaning c_{ij} is a value how much peer i trusts peer j. To obtain the c_{ij} value, we firstly need non-normalized local trust values (s_{ij}) which are calculated as the difference in the count of satisfactory $sat(i,j)$ and unsatisfactory $unsat(i,j)$ transactions between peer i and j. s_{ij} is the evaluation of transaction between i and j from the view of peer i.

$$s_{ij} = sat(i,j) - unsat(i,j)$$

The final normalized local trust value c_{ij} is then calculated using this formula.

$$c_{ij} = \begin{cases} \frac{\max(s_{ij},0)}{\sum_j \max(s_{ij},0)} & \sum_j \max(s_{ij},0) \neq 0 \\ 0 & otherwise \end{cases}$$

If there is at least one value s_{ij} in every row which is positive, all values c_{ij} will be non-negative and the sum of each row will be 1, therefore it is a row stochastic matrix or essentially a transition matrix of a Markov Chain. However, if $\sum_j \max(s_{ij}, 0) = 0$ the c_{ij} is undefined and we then set the whole row to 0 and the trust matrix is not row stochastic. This problem is solved by introducing globally pre-trusted peers. This solution is represented in Section 1.5 but for the comparison with the rest of the algorithms, there will not be any pre-trusted peers. This normalized trust matrix also does not distinguish between a peer that i had bad experience with and a peer it did not communicate with at all [5]. This issue is solved by the next algorithm called MaxTrust, which is introduced in Section 2.

Example 1.1.1. Consider a network with 3 peers. The s_{ij} values are: $s_{12} = 7,\ \ s_{21} = 5,\ \ s_{13} = 4,\ \ s_{31} = -2$ and $s_{32} = -1$ as can be seen in the graph below

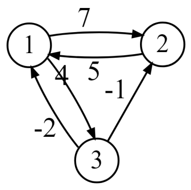

Figure 1.1: Trust graph

This graph can be written into a matrix

$$A = \begin{pmatrix} 0 & 7 & 4 \\ 5 & 0 & 0 \\ -2 & -1 & 0 \end{pmatrix}$$

Figure 1.2: Trust matrix

Firstly we apply $\max(a_{ij}, 0)$ for $i, j \in (1, 2, 3)$. Then we will calculate the sum for each row and divide each element in a row by the corresponding sum. If the sum of a row is zero (as in the last row) we set every element in this row to zero. The final normalized matrix then looks like this:

$$C = \begin{pmatrix} 0 & \frac{7}{11} & \frac{4}{11} \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Figure 1.3: Normalized trust matrix

1.2 Main idea behind EigenTrust

EigenTrust uses aggregation of the normalized local trust values (transitivity of opinions). Peer i simply weights opinions of other peers by its trust in them.

$$t_{ik} = \sum_j c_{ij} c_{jk}$$

t_{ik} is then trust that peer i has for peer k based on asking others. This can be also rewritten in matrix notation where $C = [c_{ij}]$ (normalized matrix of local trust values) and $\vec{t_i}$ is a vector of values t_{ik}.

$$\vec{t_i} = C^T \vec{c_i}$$

To get greater reach to others opinions, the multiplication continues.

$$\vec{t} = (C^T)^2 \vec{c_i} \ldots \vec{t} = (C^T)^n \vec{c_i}$$

After large number of iterations (n) the peer will have a complete view of the network and the vectors $\vec{t_i}$ for every peer i converge to the same vector under the conditions that the matrix C is irreducible and aperiodic. If the conditions are satisfied the final vector is the stationary distribution of the Markov Chain. The stationary distribution is equal to the dominant eigenvector of C. Because of the Perron-Frobenious Theorem for row stochastic matrices this eigenvector is associated with the dominant eigenvalue which absolute value is always 1 [6]. Therefore this problem can be also viewed as a solution of linear equations. Recall the definition of eigenvectors and eigenvalues is $C^T v = \lambda v$. If $\lambda = 1$ then

$$C^T v = 1v$$

and this can be further rewritten as the system of linear equations

$$(C^T - I)v = 0,$$

where the sum of coordinates of v is 1.

1.3 EigenTrust Algorithm

The basic non-distributed version of EigenTrust looks like this:

```
$\vec{t^{(0)}} = \vec{e}$
repeat
    $\vec{t^{(k+1)}} = C^T \vec{t^{(k)}}$
    $\delta = \|\vec{t^{(k+1)}} - \vec{t^{(k)}}\|$
until $\delta < \epsilon$
```

Figure 1.4: Algorithm 1 - EigenTrust

In this most basic EigenTrust version, the initial vector $\vec{t^{(0)}}$ is set to $\vec{e}$ which is a uniform probability distribution over all n peers, which means that $\vec{e_i} = 1/n$. This algorithm is based on Power iteration. The initial vector could be set to some probability distribution and the algorithm should still converge to the dominant eigenvector. The stopping condition for the cycle is that the difference between last two iterations is a small number[5].

1.4 Convergence and nonconvergence of EigenTrust

As we already mentioned the EigenTrust algorithm guarantees convergence only under some assumptions. The trust matrix has to be irreducible and aperiodic.

Definition 1.4.1. An $n \times n$ non-negative matrix T is *irreducible* if for every pair i, j of its index set, there exists a positive integer $m \equiv m(i, j)$ such that $t_{ij}^{(m)} > 0$. An irreducible matrix is said to be *cyclic (periodic)* with period d, if the period of any of its indices satisfies $d > 1$, and is said to be *acyclic (aperiodic)* if $d = 1$ [7].

To rewrite this definition in a less formal way. A Markov chain is irreducible if every node (peer) can reach every other node, it is a strongly connected graph. Whether a Markov chain is periodic or aperiodic comes in question only for irreducible Markov chains. We can calculate a period for a single node and this period will be same for others. Period of a node is a greatest common divisor (gcd) of all cycles starting in the node. As mentioned in the definition above, Markov chain is aperiodic if its period is equal to one, otherwise it is periodic.

If the normalized trust matrix is irreducible and aperiodic, the EigenTrust will converge to the dominant eigenvector with $\lambda = 1$ under the Perron-Frobenius theorem.

1.5 Pre-trusted Peers

The convergence problem and other practical issues such as having inactive peers and malicious collectives are solved in EigenTrust using a set of pre-trusted peers P. This slightly changes the structure of the algorithm, but it guarantees the convergence of the algorithm and breaks out the malicious collectives and therefore the presence of pre-trusted peers is essential to this algorithm.

```
$\vec{t^{(0)}} = \vec{p}$
**repeat**
    $\vec{t^{(k+1)}} = C^T \vec{t^{(k)}}$
    $\vec{t^{(k+1)}} = (1-a)\vec{t^{(k+1)}} + a\vec{p}$
    $\delta = \|\vec{t^{(k+1)}} - \vec{t^{(k)}}\|$
**until** $\delta < \epsilon$
```

Figure 1.5: Algorithm 2 - EigenTrust with pre-trusted peers

The initial vector change from $\vec{e}$ to $\vec{p}$ which is a distribution over P and is defined as follows:

$$p_i = \begin{cases} 1/|P| & i \in P \\ 0 & \text{otherwise} \end{cases}$$

Once we have the this distribution we can redefine the normalization of matrix.

$$c_{ij} = \begin{cases} \frac{\max(s_{ij},0)}{\sum_j \max(s_{ij},0)} & \text{if } \sum_j \max(s_{ij},0) \neq 0 \\ p_j & \text{otherwise} \end{cases}$$

The value a is some constant between 0 and 1.

Chapter 2

MaxTrust

MaxTrust [8] is another trust measuring algorithm which tries to solve the problems EigenTrust have with reducible matrices. It deals with this issue situating the EigenTrust within Max-Plus algebra [9, 10]. The main reason is the possibility of differentiating a communication of a node with inactive peers and with peers it has no connection with. For this the normalized trust matrix $C = [c_{ij}]$ is then transformed as $\overline{C} = [\bar{c}_{ij}]$ where

$$\bar{c}_{ij} = \begin{cases} c_{ij} & if(i,j) \in E \\ -\infty & \text{otherwise} \end{cases}$$

Because of the existence of $-\infty$ the Max-Plus algebra is needed.

2.1 Max-Plus Algebra

We will describe Max-Plus algebra in more informal way, to show how basic mathematical operations, needed for MaxTrust algorithm, works.

Max-Plus algebra works over $\mathbb{R}_{max} = \mathbb{R} \cup \{\epsilon\}$, where $\mathbb{R}$ is the set of real numbers and $\epsilon = -\infty$. The addition and multiplication is then defined as follows:

$$x \oplus y = \max(x, y)$$
$$x \otimes y = x + y$$

The "zero element" is $\epsilon = -\infty$ and the "unit element" is $e = 0$.

$$\epsilon \oplus a = \max(\epsilon, a) = a$$
$$\epsilon \otimes a = \epsilon + a = \epsilon$$

$$e \otimes a = e + a = a$$

The basic operations for matrices are naturally extended to Max-Plus as follows:

$$[A \oplus B]_{ij} = [A]_{ij} \oplus [B]_{ij} = \max([A]_{ij}, [B]_{ij})$$

$$[A \otimes B]_{ij} = \bigoplus_{k=1}^{n}([A]_{ik} \otimes [B]_{kj}) = \max([A]_{i1} + [B]_{1j}, \dots, [A]_{in} + [B]_{nj})$$

where n is number of columns in A and number of rows in B.

$\mathcal{E}$ is a $m \times n$ matrix where all elements are equal to ϵ. Identity matrix of dimension $n \times n$ E_n is defined as:

$$[E]_{ij} = \begin{cases} e & i = j \\ \epsilon & i \neq j \end{cases}$$

The power of a matrix $A \in \mathbb{R}_{\max}^{n \times n}$:

$$A^{\otimes^0} = E_n$$
$$A^{\otimes^k} = A \otimes A^{\otimes^{k-1}} \text{ for } k > 1$$

2.1.1 Eigenvalues and eigenvectors

Eigenvectors (scalars $\lambda \in \mathbb{R}_{\max}$) and eigenvalues (vectors $v \in \mathbb{R}_{\max}^{n}$ where $v \neq (\epsilon, \dots, \epsilon)$) in Mas-Plus for some matrix $A \in \mathbb{R}_{\max}^{n \times n}$ satisfy this equation:

$$A \otimes v = \lambda \otimes v$$

Because we will be using a Power Method similar to the one in EigenTrust, the output will be a dominant eigenvector. The eigenvalue will be obtained using

$$(A \otimes v) \otimes (-v) = \lambda.$$

You can see that lambda in this case will be a vector with all elements equal to the eigenvalue.

$$D = \begin{pmatrix} D_{11} & D_{12} & \cdots & \cdots & D_{1n} \\ \mathcal{E} & D_{22} & \cdots & \cdots & D_{2n} \\ \mathcal{E} & \mathcal{E} & D_{33} & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathcal{E} & \mathcal{E} & \mathcal{E} & \cdots & D_{nn} \end{pmatrix}$$

Figure 2.1: Normal form

2.1.2 Normal form

An essential part of Max-Trust algorithm is rewriting the trust matrix into its normal form. In Max-Plus is the normal form of a reducible matrix $A \in \mathbb{R}_{\max}^{n \times n}$ defined as a block upper triangular matrix as in Figure 2.1.

Note that in the normal form the matrix D_{nn} is irreducible and D_{ii} are either irreducible or equal to ϵ, for all $i \in (1, 2, \dots, n)$. Also note that the normal form may not be unique. The normal form can be obtained via simultaneous row/column permutation PAP^T, where P is and $n \times n$ permutation matrix.

As in EigenTrust, we will be using the transposition of the trust matrix $\overline{C}^T$ and therefore the normal form will be created from $\overline{C}^T$.

Example 2.1.1.

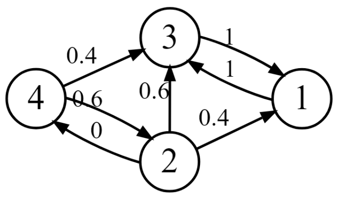

Figure 2.2: Trust Graph - Normal Form Example

$$\overline{C} = \begin{pmatrix} \epsilon & \epsilon & 1 & \epsilon \\ 0.4 & \epsilon & 0.6 & 0 \\ 1 & \epsilon & \epsilon & \epsilon \\ \epsilon & 0.6 & 0.4 & \epsilon \end{pmatrix}$$

Figure 2.3: Trust Matrix - Normal Form Example

You can see in Figure 2.2, that there are two strong conected components

$((1,3),(2,4))$. From nodes $(2,4)$ we can reach the other component $(1,3)$, but not the other way around. This examination of the trust graph is essential for creating the normal form of a matrix. In ([7]) the strongly connected components are called classes. There are two types of classes - essential and inessentail $(2,4)$ is an inessential class and $(1,3)$ is an essential class.

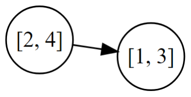

Figure 2.4: Reduced Trust Graph

For better visualization of essential and inessential classes we can reduce the graph as in Figure 2.4. The essential classes are terminal states in the reduced graph.

Finally the trust matrix in Figure 2.3 can be converted to the normal form using the permutation matrix generated from the permutation of nodes $((1,2,3,4) \mapsto (2,4,1,3))$. This permutation is obtained by firstly taking the inessential classes and sorting them accordingly (earlier occuring classes cannot be reached from later occuring classes). The essential classes follows the inessential classes.

The normal form for MaxTrust is created from $\overline{C}^T$ which means that we can look at the the trust graph as with switched directions of edges. This leads to swapping of classes (inessential become essential and the other way around). The final matrix $D \equiv \overline{C}^T$ with the permutation of nodes as $(1,3,2,4)$:

$$D = \begin{pmatrix} \epsilon & 1 & 0.4 & \epsilon \\ 1 & \epsilon & 0.6 & 0.4 \\ \epsilon & \epsilon & \epsilon & 0.6 \\ \epsilon & \epsilon & 0 & \epsilon \end{pmatrix}$$

Figure 2.5: Trust Matrix in Normal Form

The blocks are then:

$$D_{11} = \begin{pmatrix} \epsilon & 1 \\ 1 & \epsilon \end{pmatrix} \qquad D_{22} = \begin{pmatrix} \epsilon & 0.6 \\ 0 & \epsilon \end{pmatrix} \qquad D_{12} = \begin{pmatrix} 0.4 & \epsilon \\ 0.6 & 0.4 \end{pmatrix}$$

Figure 2.6: Normal form blocks

2.2 Power Method and MaxTrust

This thesis will not cover the full theoretical background of MaxTrust algorithm. The main idea is to compute a generalized eigenmode of a reducible trust matrix through calculating the eigenvalues and eigenvectors of irreducible blocks of the matrix found on the diagonal of the normal form of the matrix using the Power Method. The Power Method (Power iteration) is used in the most basic implementation of the EigenTrust algorithm. As in EigenTrust, the calculation uses the transposition of the trust matrix $\overline{C}$ for the aggregation of opinions.

```
$p = 0$
$v_p = r$
**repeat**
    $v_{p+1} = \overline{C}^T \otimes v_p$
    $p = p + 1$
**until** $(\exists q)(v_q = c \otimes v_p \text{ and } c \geq 0)$
$\lambda = \frac{c}{p-q}$
$v = \bigoplus_{i=1}^{p-q}(\lambda^{\otimes^{p-q-i}} \otimes v_{q+i-1})$
**return** $(\lambda, v)$
```

Figure 2.7: Algorithm 3 - *max_power*

This procedure is by the authors of MaxTrust called *max_power.* Where the input is an irreducible trust matrix $\overline{C}$ and arbitrary vector of trust values r. Note that the official implementation uses the standard algebra multiplication ($\overline{C}^T v_p$) but other papers discussing this matter use the Max-Plus multiplication [9, 10]. The Power Method computes the dominant eigenvector and eigenvalue of a matrix, so when using the transposition as in the *max_power*, it computes the eigenvalue and eigenvector for the transposed matrix. In this thesis we will use the Max-Plus algebra multiplication, because when using the standard one, the output after few iterations will be always a vector with all elements equal to $-\infty$, because there are always at least $-\infty$ on the diagonal.

The *max_power* procedure can be used only for irreducible matrices. If used on reducible matrices its convergence is uncertain. Therefore for the reducible cases, the MaxTrust algorithm is introduced. However, the MaxTrust algorithm does not calculate the eigenvector, but it is similar to the eigenmode computation of the trust matrix $\overline{C}^T$ [9, 11, 12]. There are some slight differences with the cited papers, but the idea is similar.

Generalized eigenmode

A pair of vectors $(\eta, v) \in \mathbb{R}^n \times \mathbb{R}^n$ is called a *generalized eigenmode* [9] of the regular matrix A if for all $k \geq 0$

$$A \otimes (\eta^{\otimes^k} \otimes v) = \eta^{\otimes^{k+1}} \otimes v$$

or rewritten

$$A \otimes (\eta^{\otimes^k} \otimes v) = \eta \otimes (\eta^{\otimes^k} \otimes v).$$

Generalized eigenmode (or simply eigenmode) shows periodic behaviour of a system.

The MaxTrust algorithm is then defined as follows:

```
$D = get\_normal\_form(\overline{C})$
$\lambda_n, v_n = max\_power(D_{nn}, w_n)$
$\xi_n = \lambda_n$
j = n - 1
while $j \geq 1$
    $\lambda_j = max\_power(D_{jj}, w_j)$
    if $\lambda_j > \xi_{j+1}$ then
        $\xi_j = \lambda_j$
        $v_j = \bigoplus_{k=1}^{n} D_{jk} \otimes w_k \otimes \lambda_j^{\otimes^{j-1}}$
    else
        $\xi_j = \lambda_{j+1}$
        $v_j = (\xi_j)^{-1} \bigoplus_{k=1}^{n} D_{jk} \otimes w_k \otimes \lambda_j^{\otimes^{j-1}}$
    $j = j - 1$
return $t = v \otimes \xi^{\otimes^T}$
```

Figure 2.8: Algorithm 4 - MaxTrust

This procedure takes as input a regular reducible trust matrix C, a unitary vector of initial trust values w and a terminal time T corresponding to the k in the definition of generalized eigenmode. It returns a vector of trust values t at the terminal time. Note that $\eta = \xi$ and the terminal time T corresponds to k in the definition of eigenmode. Also note that for the algorithm to work properly, the trust matrix has to be regular which means that every row must contain at least one element different from ϵ [12].

Example 2.2.1. Let's continue with the graph in Example 2.1.1. We already have the normal form of the trust matrix. There are two diagonal blocks, so we set $n = 2$. We will also set the initial trust vector $w = (0.25, 0.25, 0.25, 0.25)$.

Firstly using the *max_power* procedure we will calculate the λ_2 and v_2 of the last diagonal block (D_{22} in Figure 2.6) with the input trust vector $w_2 = (0.25, 0.25)$. We obtain $\lambda_2 = 0.3$ and $v_2 = (0.85, 0.25)$. We set the $\xi_2 = 0.3$.

$$D = \begin{pmatrix} \epsilon & 1 & 0.4 & \epsilon \\ 1 & \epsilon & 0.6 & 0.4 \\ \epsilon & \epsilon & \epsilon & 0.6 \\ \epsilon & \epsilon & 0 & \epsilon \end{pmatrix}$$

Figure 2.9: Normal form

We can now set $j = n - 1 = 1$, therefore we have only one iteration left of the main cycle. After calculating $\lambda_1 = 1$ from the first diagonal block with $w_1 = w_2$, we can check the condition. $1 = \lambda_1 > \xi_2 = 0.3$, therefore we can set $\xi_1 = 1$ and calculate the $v_1 = \bigoplus_{k=1}^{2} D_{1k} \otimes w_k \otimes \lambda_1^{\otimes^0}$. Because the neutral element to the Max-Plus multiplication is 0, the term $\lambda_1^{\otimes^0}$ will be 0 as well.

$$D_{11} \otimes w_1 = \begin{pmatrix} \epsilon & 1 \\ 1 & \epsilon \end{pmatrix} \otimes \begin{pmatrix} 0.25 \\ 0.25 \end{pmatrix} = \begin{pmatrix} 1.25 \\ 1.25 \end{pmatrix}$$

$$D_{12} \otimes w_2 = \begin{pmatrix} 0.4 & \epsilon \\ 0.6 & 0.4 \end{pmatrix} \otimes \begin{pmatrix} 0.25 \\ 0.25 \end{pmatrix} = \begin{pmatrix} 0.65 \\ 0.85 \end{pmatrix}$$

$$v_1 = \bigoplus(\begin{pmatrix} 0.65 \\ 0.85 \end{pmatrix}, \begin{pmatrix} 1.25 \\ 1.25 \end{pmatrix}) = \begin{pmatrix} \max(0.65, 1.25) \\ \max(0.85, 1.25) \end{pmatrix} = \begin{pmatrix} 1.25 \\ 1.25 \end{pmatrix}$$

As we have the whole vector $v = (1.25, 1.25, 0.85, 0.55)$, we need to apply the terminal time. First we will set $T = 1$. In this context, we take $\xi = (1, 1, 0.3, 0.3)$. The final vector t is then

$$t_1 = v \otimes \xi^{\otimes^1} = \begin{pmatrix} 1.25 \\ 1.25 \\ 0.85 \\ 0.55 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 1 \\ 0.3 \\ 0.3 \end{pmatrix} = \begin{pmatrix} 2.25 \\ 2.25 \\ 1.15 \\ 0.85 \end{pmatrix},$$

which is our final trust vector. The output in time $T = 10$ for example is

$$t_{10} = v \otimes \xi^{\otimes^{10}} = \begin{pmatrix} 1.25 \\ 1.25 \\ 0.85 \\ 0.55 \end{pmatrix} \otimes \begin{pmatrix} 10 \\ 10 \\ 3 \\ 3 \end{pmatrix} = \begin{pmatrix} 11.25 \\ 11.25 \\ 3.85 \\ 3.55 \end{pmatrix},$$

Recall that the normal form is permutated original matrix, so we need to bring those values to the original permutation ($[1, 2, 3, 4]$). t in time $T = 1$ is $t_1 = (2.25, 1.15, 2.25, 0.85)$ and in time $T = 10$ is $t_{10} = (11.25, 11.25, 3.85, 3.55)$.

Let's check if the eigenmode definition for our output holds. Note that the normal form was created from the transposition of $\overline{C}$, therefore we were computing eigenvectors for the $\overline{C}^T$. Recall that $\eta = \xi$ and $t_1 = \eta^{\otimes^k} \otimes v$

$$\overline{C}^T \otimes t_1 = \begin{pmatrix} \epsilon & 0.4 & 1 & \epsilon \\ \epsilon & \epsilon & \epsilon & 0.6 \\ 1 & 0.6 & \epsilon & 0.4 \\ \epsilon & 0 & \epsilon & \epsilon \end{pmatrix} \otimes \begin{pmatrix} 2.25 \\ 1.15 \\ 2.25 \\ 0.85 \end{pmatrix} = \begin{pmatrix} 3.25 \\ 1.45 \\ 3.25 \\ 1.15 \end{pmatrix} = \begin{pmatrix} 1 \\ 0.3 \\ 1 \\ 0.3 \end{pmatrix} \otimes \begin{pmatrix} 2.25 \\ 1.15 \\ 2.25 \\ 0.85 \end{pmatrix} = \eta \otimes t_1$$

2.2.1 Normalization of eigenvector/eigenmode

If a pair (η, v) creates a generalized eigenmode of a matrix, also the pair $(\eta, \alpha \otimes v)$ does for any $\alpha \in \mathbb{R}$ [9]. Which means that we can add or subtract a real number α from every element of vector v. This operation can be used on the eigenvector of an irreducible matrix as well. We cannot always obtain a vector, which sum of coordinates is 1, but we can subtract some value, for a better comparison with other algorithms.

2.2.2 Summary

The MaxTrust algorithm can be used for irreducible and reducible trust matrices which solves some issues EigenTrust has. However, there is still the condition of regularity of the trust matrix and the output not being an eigenvector, but similar to eigenmode, for reducible matrices.

Chapter 3

SHAPE-Trust

SHAPE-Trust [4] is another trust management system with a very different approach on calculating the global trust values from the previous two methods. This algorithm uses knowledge of Game Theory, more precisely the application of Coalitional games and Shapley value.

Game Theory is a study of mathematical applications to model and analyze situations of interactive decision making of some "players" with different goals. The game theory is divided into noncooperative (strategic) and cooperative (coalitional) games. In strategic games the players make decisions independently. In coalitional games the players can form groups (coalitions) that make decision based on an agreed strategy. Shapley value [13] is a solution concept for coalitional games. It represents the fair and efficient payoff of a player for participating in a game. In the case of P2P networks it will represent the global trust assigned to each peer.

3.1 Model design of SHAPE-Trust

3.1.1 Trust game

The SHAPE-trust is defined as a coalitional game of peers as player. A coalition is a subset S of all peers N ($S \subseteq N$). The first problem is to find a suitable definition of the game (trust game). In game theory, a coalitional game is a function mapping over the set of all possible coalitions to a real number

$$v : \mathcal{P}(N) \rightarrow \mathbb{R},$$

where $\mathcal{P}(N) = \{A | A \subseteq N\}$ is called the powerset and $v(\emptyset) = 0$. The trust game uses the decomposition into internal and external trust. The internal trust is generated as the total intercoalitional trust among the peers of a coalition S.

The external trust is the trust given to the members of S from peers outside of this coalition and is based on the pessimistic evaluation. These two parts are combined additively. The trust game is defined as follows.

Definition 3.1.1. Let $\mathcal{G} = (N, E)$ be a trust graph. For every coalition of peers $S \subseteq N$, define

$$S^* = \{j \in S | \text{there exists } i \notin S \text{ such that } (i, j) \in E\}$$

The trust game $v_{\mathcal{G}}$ is given by

$$v_{\mathcal{G}}(S) = \sum_{\substack{i,j \in S \\ (i,j) \in E}} a_{ij} + \sum_{j \in S^*} \min_{\substack{i \notin S \\ (i,j) \in E}} a_{ij}, \qquad S \subseteq N$$

As can be seen in the Definition 3.1.1 if $S^* = \emptyset$, the second summand is equal to zero by the definition of empty sum. This property of $v_{\mathcal{G}}$ is essential to ensure that the trust game is indeed a coalitional game. The first summand sums up the trust values of peers inside the coalition, which represents the internal trust. The second summand calculates the pessimistic evaluation of the external trust using the minimum function. This sum takes all peers in S that are directly rated by at least one peer from outside the coalition (S^*) and adds the minimal trust for each of them. “The use of minimum instead of, say, multiplication, avoids paradoxical situations when the peer highly trusted by many other peers would have very low trust.”[4] The trust game also has basic properties of coalitonal games:

Monotony A coalitional game v is called monotone if

$$v(S) \leq v(T)$$

for all $S, T \in \mathcal{P}(N)$ with $S \subseteq T$.

Supperadditivity A coalitional game v is called superadditive if

$$v(S) + v(T) \leq v(S \cup T)$$

for all $S, T \in \mathcal{P}(N)$ with $S \cap T = \emptyset$.

For the proof of these properties see ([4] Preposition 3.3).

Example 3.1.1. Let's take the same graph and its normalized trust matrix as in Example 1.1.1. The graph $\mathcal{G} = (N, E)$ after normalization looks like this.

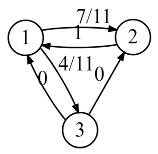

Figure 3.1: Trust matrix

Now let's calculate the trust game for every possible coalition in the powerset $\mathcal{P}(N) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{2,3\}, \{1,3\}, \{1,2,3\}\}$.

$$v_{\mathcal{G}}(\emptyset) = 0$$
$$v_{\mathcal{G}}(\{1\}) = 0 + \min(1, 0) = 0$$
$$v_{\mathcal{G}}(\{2\}) = 0 + \min(7/11, 0) = 0$$
$$v_{\mathcal{G}}(\{3\}) = 0 + \min(4/11) = 4/11$$
$$v_{\mathcal{G}}(\{1,2\}) = (7/11 + 1) + \min(0) + \min(0) = 18/11$$
$$v_{\mathcal{G}}(\{2,3\}) = 0 + \min(7/11) + \min(4/11) = 1$$
$$v_{\mathcal{G}}(\{1,3\}) = (4/11 + 0) + \min(1) = 15/11$$
$$v_{\mathcal{G}}(\{1,2,3\}) = 7/11 + 4/11 + 1 + 0 + 0 = 2$$

3.1.2 Solution concept and Shapley value

Once the trust game is established a solution concept for this game is needed. The *Shapley value* for a coalitional game v refers to the allocation $\phi(v) \in \mathbb{R}^N$, where each coordinate is calculated as follows.

$$\phi_i(v) = \sum_{S \subseteq N \setminus i} \frac{|S|!(n - |S| - 1)!}{n!} \cdot \Delta_i v(S) \qquad i \in N$$

where

$$\Delta_i v(S) = v(S \cup i) - v(S)$$

is the marginal contribution of player $i \notin S$ to coalition S and the first factor is the probability distribution p_i where

$$\sum_{S \subseteq N \setminus i} p_i(S) = 1.$$

The Shapley value has the null player property and is symmetric, efficient and additive. Those features are called the basic axioms of fairness [13].

Null player property

A player is called a null player in a game v if

$$v(A \cup i) = v(A), \qquad \text{for all } A \subseteq N \setminus i.$$

The null player property is satisfied if for each game v and each player $i \in N$ the implication

$$i \text{ is a null player in } v \qquad \Longrightarrow \qquad \phi_i(v) = 0.$$

holds.

Symmetry

Players $i, j \in N$ are symmetric in a game v if

$$v(A \cup i) = v(A \cup j), \qquad \text{for each coalition } A \subseteq N \setminus ij.$$

The allocation is symmetric if for each game v and all players $i, j \in N$ the following implication holds:

$$i \text{ and } j \text{ are symmetric} \qquad \Longrightarrow \qquad \phi_i(v) = \phi_j(v).$$

Efficiency

The allocation value is efficient when

$$\phi_1(v) + \cdots + \phi_n(v) = v(N) \qquad \text{for every game } v.$$

Additivity

$$\phi(u + v) = \phi(u) + \phi(v) \qquad \text{for every two games } u, v.$$

Linearity

$$\phi(\alpha u + \beta v) = \alpha\phi(u) + \beta\phi(v) \qquad \text{for every two games } u, v \text{ and all } \alpha, \beta \in \mathbb{R}.$$

In a more informal way, the Shapley value is a form of calculation of the contribution of a single peer to all possible coalitions and therefore tells how much is the peer valueable, or in the meaning of the trust game, how much is the peer trusted.

Example 3.1.2. Let's calcualte the Shapley value for trust game in Example 3.1.1.

For player 1 the all possible $S \subseteq N \backslash i$ are $\{\emptyset, \{2\}, \{3\}, \{2,3\}\}$.

$$p_1(\emptyset) = \frac{0!(3-0-1)!}{3!} = \frac{2}{6} \qquad \Delta_1 v(\emptyset) = v(\{1\}) - v(\emptyset) = 0$$

$$p_1(\{2\}) = \frac{1!(3-1-1)!}{3!} = \frac{1}{6} \qquad \Delta_1 v(\{2\}) = v(\{1,2\}) - v(\{2\}) = 18/11$$

$$p_1(\{3\}) = \frac{1!(3-1-1)!}{3!} = \frac{1}{6} \qquad \Delta_1 v(\{3\}) = v(\{1,3\}) - v(\{3\}) = 15/11 - 4/11 = 1$$

$$p_1(\{2,3\}) = \frac{2!(3-2-1)!}{3!} = \frac{2}{6} \qquad \Delta_1 v(\{2,3\}) = v(\{1,2,3\}) - v(\{2,3\}) = 2 - 1 = 1$$

The Shapley value for player one is

$$\phi_1 = \frac{2}{6} \cdot 0 + \frac{1}{6} \cdot \frac{18}{11} + \frac{1}{6} \cdot 1 + \frac{2}{6} \cdot 1 = \frac{51}{66} \approx 0.7727.$$

The Shapley value for player 2 and 3 are calculated using the same formula:

$$\phi_2 \approx 0.5909 \qquad \phi_3 \approx 0.6363$$

There are no null players nor symmetric players and we are using just single game (trust game) so we do not need to check the additivity or linearity, but we can check the efficiency property:

$$\phi_1 + \phi_2 + \phi_3 = 2 = v(\{1,2,3\}).$$

Chapter 4

Numerical Experiments

In this section you will find numerical experiments which focus on different scenarios of P2P networks and especially on the reducible cases. For all of the following experimental scenarios the trust graphs were artificially created to exhibit the different properties of theirs trust matrices. We will compare the ranking of outputs of all three previously presented algorithms. In every scenario we will assume, that there are no pre-trusted peers. This may create a problematic environment for the EigenTrust algorithm. We will examine only small networks of 3, 4, 5 and 10 peers. Every example will be presented with its trust graph and/or trust matrix, to visualize the problematic aspects of each example. The global trust vectors of each algorithm were calculated by its implementation in Julia programming language (link in appendix). All MaxTrust computations were done with the input vector w set to a uniform distribution vector and the time T set to 1. EigenTrust's initial vector was a uniform distribution as well.

4.1 Experiment #1

The first experiment is based on the running Example 3.1.1. In this example the malicious peer is 3. Peer-3 obtained a worse trust from 1, than 2 did and intentionally set its trust for other peers low (after normalization it is 0). The trust matrix of this network as the input for EigenTrust is

$$C = \begin{pmatrix} 0 & \frac{7}{11} & \frac{4}{11} \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Figure 4.1: Trust matrix - Exp #1

,which is a reducible matrix from the point of view of regular algebra. We can calculate the eigenvectors and eigenvalues of this matrix, but since the prerequisities for Perron-Frobenious theorem are not satisfied, there will not be a dominant eigenvalue with value 1.

If we transform C to the Max-Plus algebra

$$\overline{C} = \begin{pmatrix} \epsilon & \frac{7}{11} & \frac{4}{11} \\ 1 & \epsilon & \epsilon \\ 0 & 0 & \epsilon \end{pmatrix}$$

Figure 4.2: Max-Plus Trust matrix

the matrix becomes regular and irreducible according to this algebra, meaning the *max_power* procedure is sufficient for calculating the eigenvector. The EigenTrust's Power iteration in this experiment returned a vector $(0, 0, 0)$, therefore, there are only zeros in Figure 4.3, where there should be blue bars. In the graph below we subtracted 1.5 from the output of MaxTrust.

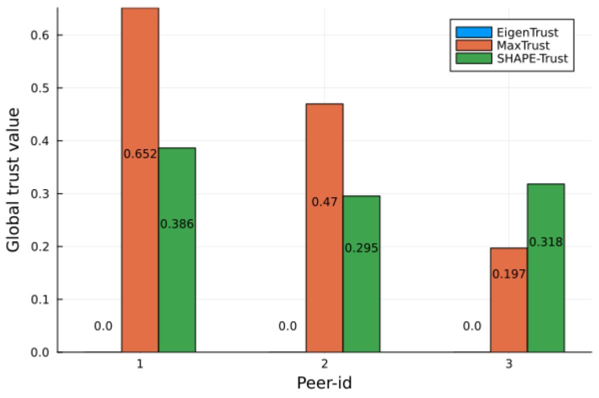

Figure 4.3: Trust - Exp #1

In Figure 4.3, you can see, that MaxTrust better distributed the trust. But our main focus is on the ranking of nodes.

algorithm	peer-1	peer-2	peer-3
MaxTrust	1.	2.	3.
SHAPE-Trust	1.	3.	2.

Table 4.1: Ranking - Exp #1

Both algorithms ranked peer-1 as the most trusted as it rationally should be, but unfortunately SHAPE-Trust ranked peer-3 as the second most trusted, which is not ideal.

4.2 Experiment #2

The second experiment is based on Example 2.2.1. The malicious peer is peer-4. He did not rated intentionally badly, but he has been sending malicious files to peer-2.

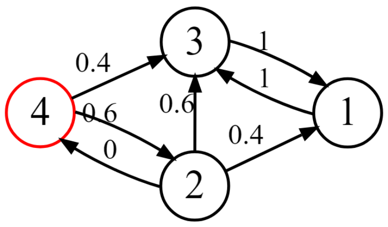

Figure 4.4: Graph - Exp #2

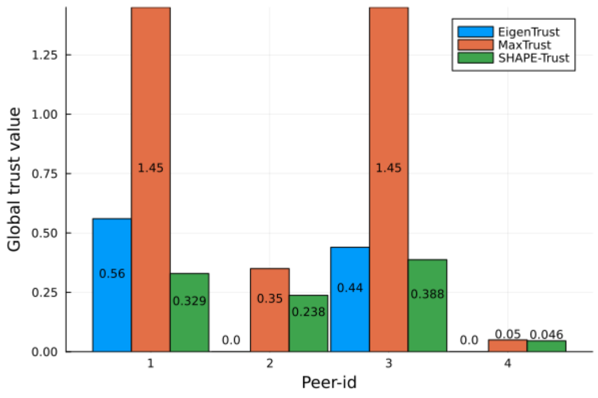

Figure 4.5: Trust - Exp #2

As can be seen in the Figure 4.5 and Table 4.2. All algorithms recognized that peer 4 is malicious.

algorithm	peer-1	peer-2	peer-3	peer-4
EigenTrust	1.	3.	2.	4.
MaxTrust	1.	3.	2.	4.
SHAPE-Trust	2.	3.	1.	4.

Table 4.2: Ranking - Exp #2

All algorithm rated the peers appropriately.

4.3 Experiment #3

In the third experiment we have five nodes and malicious peer was marked peer-3. Peer-3 in this network intentionally gave low trust to peers-4 and peer-5 and has been sending mix of fine and corrupted files to the peer-1 and peer-2. The Figure 4.6 contains already normalized trust values.

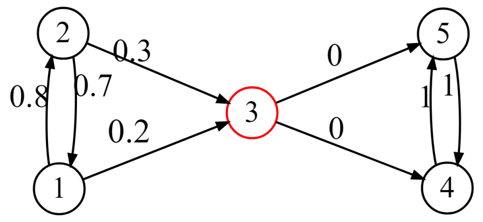

Figure 4.6: Trust Graph - Exp #3

algorithm	peer-1	peer-2	peer-3	peer-4	peer-5
EigenTrust	3.	4.	5.	1.	2.
MaxTrust	4.	3.	5.	1.	2.
SHAPE-Trust	4.	1.	5.	2.	3.

Table 4.3: Ranking - Exp #3

As can be seen in Figure 4.6, EigenTrust kind of failed again, as expected. Both Maxtrust and SHAPE-Trust recognized the malicious peer, but they ranked them quite differently. However, for the sake of security of the network, both rankings are fine.

In Figure 4.6 can be seen one the property of symmetric peers, in the trust distribution of peer-4 and peer-5.

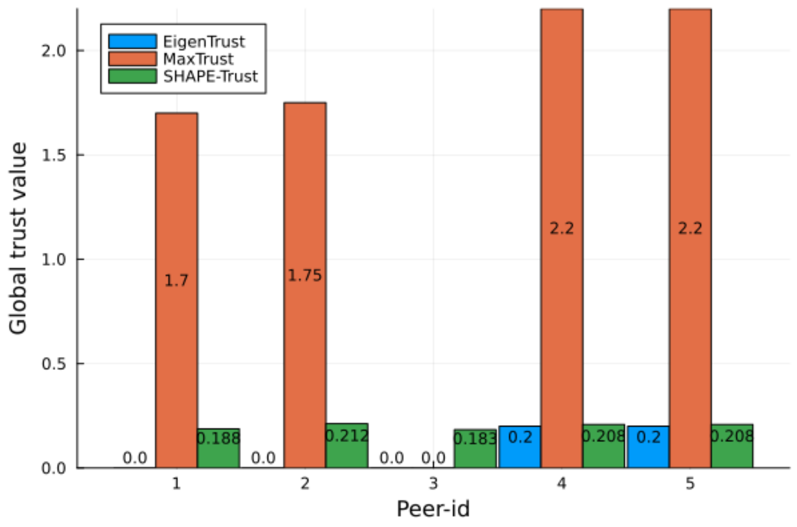

Figure 4.7: Trust - Exp #3

4.4 Experiment #4

In this experiment we have two groups, the smaller one containing four nodes is malicious and the bigger one is good. The malicious peers were sending some corrupted files to some peers from the other group, but have not been receiving any files. Therefore, there are local trust values assigned only in one way between these groups.

algorithm	peer-1	peer-2	peer-3	peer-4	peer-5	peer-6	peer-7	peer-8	peer-9	peer-10
EigenTrust	3.	7.	6.	4.	10.	8.	5.	9.	2.	1.
MaxTrust .	7.	8.	9.	10	1.	4.	5.	6.	3.	2.
SHAPE-Trust	5.	10	7.	4.	8.	3.	6.	9.	2.	1.

Table 4.4: Ranking - Exp #4

As can be seen in the Table 4.4 and Figure 4.8, MaxTrust dominated in recognizing the two groups of good and malicious peers. SHAPE-Trust and EigenTrust in this scenario ranked the peers quiet similarly. The ranking of SHAPE-Trust, again for the sake of security, is quite fine, when the the first three nodes are from the group of good peers.

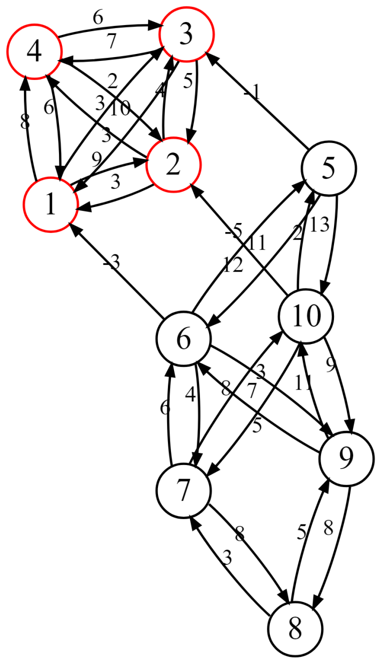

Figure 4.8: Trust Graph - Exp #4

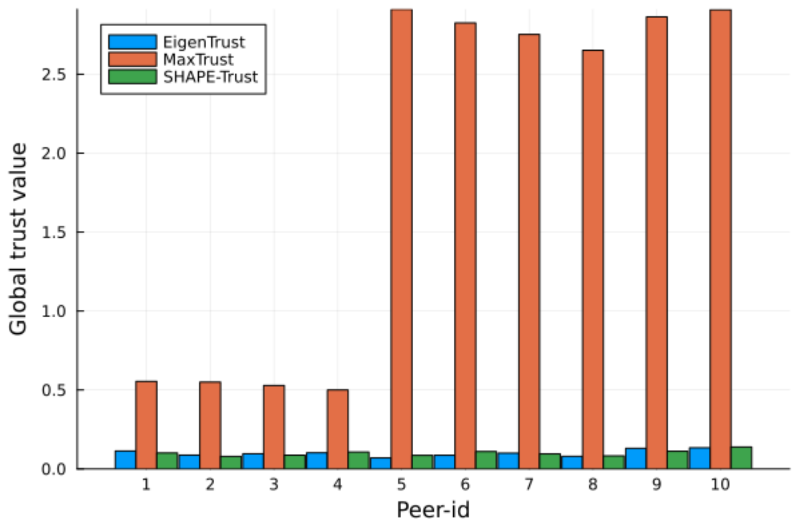

Figure 4.9: Trust - Exp #4

Conclusions

This thesis focused on comparing three Trust Management Systems for P2P networks: EigenTrust, MaxTrust and SHAPE-Trust. The main point was to compare only EigenTrust and SHAPE-Trust, but we look at the MaxTrust algorithm as an expansion of EigenTrust. The first objective was implementing these algorithms in Julia programming language. Link to the GitHub repository can be found in the appendix. The algorithms are described with their basic theoretical background in the first three chapters. Chapter 4 focuses on the comparison of these algorithms in simulated experiments, mainly on reducible cases, which can create a problem for EigenTrust. These experiments showed, that for a network without pre-trusted peers and with, for example, only one way communication between some peers, it creates a problematic environment for the EigenTrust. SHAPE-Trust, however, acts quite well in all situations, with fairly distributing the trust between all peers and with prioritizing the good peers over the malicious ones. The MaxTrust algorithm, as can be seen in the experiments, usually behaved well, but its implementation is quite challenging, due to many parameters of the original algorithm.

References

[1] Aberer K, Hauswirth M (2002) An overview of peer-to-peer information systems. In: WDAS

[2] Hendrikx F, Bubendorfer K, Chard R (2015) Reputation systems: A survey and taxonomy. Journal of Parallel and Distributed Computing 75:184–197. https://doi.org/10.1016/j.jpdc.2014.08.004

[3] Novotný M Trust management systems in P2P networks. PhD thesis, Charles University in Prague

[4] Bandhana A, Kroupa T, García S (2024) Trust in shapley: A cooperative quest for global trust in P2P networks. In: AAMAS 2024. IFAAMAS

[5] Kamvar SD, Schlosser MT, Garcia-Molina H (2003) The eigentrust algorithm for reputation management in p2p networks. In: WWW 2003

[6] Meyer CD (2000) Matrix analysis and applied linear algebra

[7] Seneta E (1981) Non-negative matrices and markov chains

[8] Afanador J, Oren N, Baptista M, Araujo M (2020) From eigentrust to a trust-measuring algorithm in the max-plus algebra. ECAI 2020

[9] Heidergott B, Olsder GJ, Woude J van der (2006) Max plus at work: Modeling and analysis of synchronized systems: A course on max-plus algebra and its applications. Princeton University Press

[10] Al Bermanei H (2021) Applications of max-plus algebra to scheduling. PhD thesis

[11] Mursyidah H, Subiono N (2017) Eigenvalue, eigenvector, eigenmode of reducible matrix and its application. AIP conference proceedings. https://doi.org/10.1063/1.4994447

[12] Konigsberg ZR (2009) A generalized eigenmode algorithm for reducible regular matrices over the max-plus algebra. International Mathematical Forum 4(24):1157–1171. https://doi.org/10.1109/ccdc.2009.5195195

[13] Maschler M, Solan E, Zamir S (2013) Game theory. Cambridge University Press

Appendix

Link to the GitHub repository with algorithm implementations

https://github.com/ruttejan/TMS