Defining Trustworthiness in Signed Networks

Dhruv Patel
Indian Institute of Technology Gandhinagar
patel.dhruv@iitgn.ac.in

Haikoo Khandor
Indian Institute of Technology Gandhinagar
haikoo.ashok@iitgn.ac.in

Madhav Kanda
Indian Institute of Technology Gandhinagar
madhav.kanda@iitgn.ac.in

Abstract

The prevalence of online trade has increased as a result of recent technology developments. However, this has sparked worries about theft and fraud, particularly on websites where users' identities are kept secret, like Bitcoin OTC. It has become vital to gauge user reputations and comprehend how new users affect existing communities in order to mitigate this risk. The who-trusts-whom signed network can show both good and bad user relationships. Due to the dynamic nature of the trust levels in such networks, they are susceptible to link and node failures. The goal of this research is to create a community trustworthiness metric that minimises shifts in community trust. We thereby try to make an understanding of the network's stability with respect to the trustworthiness.

2012 ACM Subject Classification Data Science

Keywords and phrases Social Network, Trustworthiness, Weighted Signed Networks, Bitcoin OTC, Correlation Clustering, Goodness, Fairness

Acknowledgements We want to thank Prof. Anirban Dasgupta, Computer Science and Engineering, IIT Gandhinagar for his help and support throughout the project.

1 Introduction

The entire code for this project can be found in this repository:
`https://github.com/dhruvpatel144/DS_project.git`

The introduction of cryptocurrencies and blockchain technology has completely changed how we conduct financial transactions in the modern digital age. One such cryptocurrency is Bitcoin, which processes transactions safely and quickly using a decentralised network of nodes, edges, and graphs. A chain of transactions is created by joining the nodes of a graph that each transaction on the Bitcoin network is represented as a node in. A further degree of security is added to the network by weighting these edges in accordance with the amount of computation required to validate the transaction. This weight-based validation technique ensures the network's dependability and aids in preventing fraud.

The Bitcoin network is a desirable alternative for financial transactions due to the openness and security it offers, and its influence extends far beyond the realm of finance.

Problem Statement

Given the weights of a signed network, our task is to devise a metric denoting the trustworthiness of a community within a network. Further, we establish a relation between the trustworthiness metric value for each cluster and the corresponding change in the predicted weights upon the introduction of a new node.

2 Dataset

The dataset is taken from soc-sign-bitcoin-otc [1]of Snap library[2] from Stanford. This dataset was created as a part of the paper, Edge Weight Prediction in Weighted Signed Networks.

The Bitcoin OTC dataset consists of (source, target, weight, Timestamp) four columns which depict the directed edge from source to target at the timestamp specified that have been collected over a span of five years for the Bitcoin OTC network.

The dataset consists of network characteristics like:

- Source node
- Target node
- Weight
- Timestamp

Dataset Statistics	
Nodes	5881
Edges	35592
Range of edge weight	-10 to +10
Percentage of positive edges	89%

The weight represents a positive or negative relation based on positive or negative weight respectively. This may denote relationships between individuals, groups, or entities with positive and negative aspects.

	Edge weights	Frequency
0	-1	601
1	-2	182
2	-3	91
3	-4	27
4	-5	179
5	-6	5
6	-7	14
7	-8	31
8	-9	20
9	-10	2413
10	0	no Nodes with zero weight
11	1	20048
12	2	5562
13	3	2561
14	4	967
15	5	1268
16	6	265
17	7	208
18	8	277
19	9	108
20	10	765

Table 1 Table for frequency of each node

We use a small portion of the dataset for initial graph. Thereafter we generate a few new data points which correspond to new users which have entered our network.

	Type of Weights	Mean	Std
0	Positive	1.96	1.86
1	Negative	-7.55	3.72

Table 2 Table for mean and std for Positive and Negative

2.1 Dataset Preprocessing

Normalising the weights

The dataset inherently has weights defined in the range of -10 to 10(excluding 0). On the Bitcoin OTC network, according to their guidelines, a -10 rating should be given to fraudsters whereas +10 should be given to those whom you trust the way you trust yourself. Intermediate values have meaning between these. The edge weights are scaled between -1 and 1.

Normalising the data

We normalize the data by subtracting each edge weight from the mean of the edge weights. This enables the feature to be on a similar scale. In addition to this the -1 to 1 threshold is not strictly true now due to weight normalisation.

2.2 Analysis of Data

Analyzing the data helps us get insights into the data and make inferences. The timestamp plays an important role in understanding the temporal effect of users rating each other. We first analyse wrt the source or target node. The reference changes as we go from rater(source) to ratee(target).

In order to develop the intuition for fraudulent user, we make use of the definition given by Bitcoin OTC network where they recommend giving a rating of -10 for frauds. We first plot the number of ratings given by users wrt timestamp. The following plots show the number of ratings vs timestamp for the complete dataset, only positively weighted edges and negatively weighted edges.

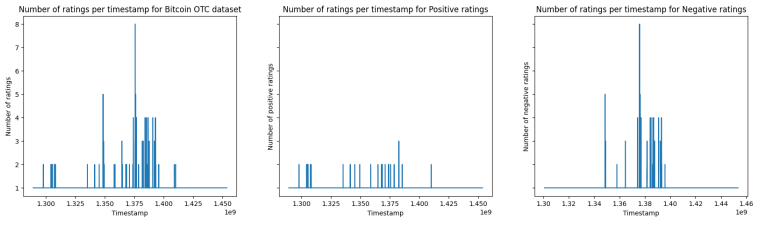

Figure 1 Number of rating - total, positive and negative wrt timestamp

From the first look, it may seem that the negative ratings are added at the same time and can be thought of an instance of fraudulent activity. For that we validate by looking from source and target reference:

2.2.1 Source:

Thus for fraudulent user detection, we first consider the nodes which may send high negative ratings(-10) at the same timestamp to multiple users. This arises from the fact that fraudsters often try to affect the rating of other users usually at the same time like spam mails. We first find the number ratings which are -10 for all timestamps.

On printing the head of the first ten entries, we associate Source nodes which are giving these ratings at a particular timestamp. Note that here fraudulent activity is talked with respect to time. Hence we evaluate them for particular timestamp. Overall a user may not be a fraud when we talk wrt the entire time frame. For most frequent timestamps, we get only single users with the number of times they have assigned -10 to the different target nodes.

But the important thing to note here is when we analyse the network in real time, for the recent timeframe. Detection of a possible fradulent activity is more important than missing a positive fraud activity.

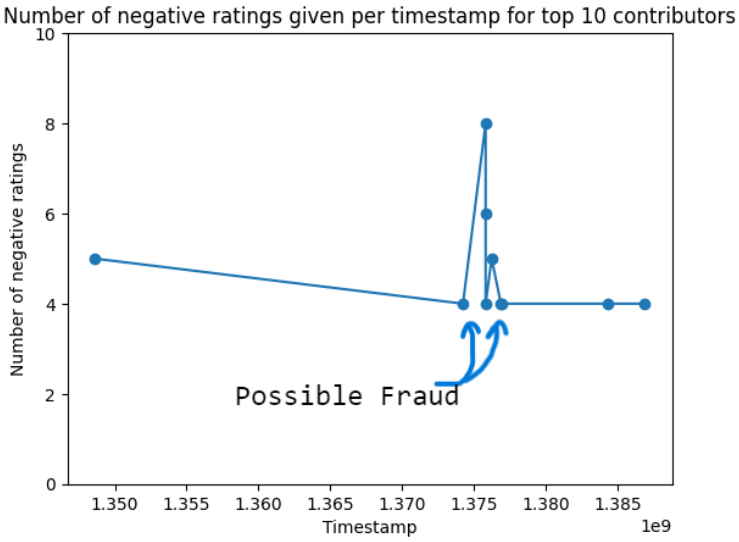

Figure 2 Fraud Detection

2.2.2 Target:

For the target, we need to make it time independent since users receive ratings from other users and to make observations, we need to evaluate across a time frame. Keeping it similar to -10 for negative intention, we define +10 as the weight for positive intentions. Here we evaluate across timeframe and find the nodes with most positive(+10) edges coming in and most negative edges also coming in.
Since the contribution of rating depends on both positive and negative, we average out the total rating adding the weights and dividing by the total incoming edges for a particular node. The reason for finding the nodes with most positive/negative edges is to account for a general observation. This is because of not to be biased by nodes with less incoming edges. We can infer the following from the table1 and table2:

1. Although the number of edge weights are greater in number for positive weights, the mean of positive weights is 1.96 whereas for negative weights, it is -7.55. This shows that negatively weighted edges have a considerate influence despite being low in number.
2. A large number of positive edges have small weights namely 1, 2 or 3. The largest weight(+10) has a frequency of 765 compared to 2413 for largest negative weighted edges(-10). This shows us that rating users positively is rare since maximum of positive ratings lie as 1,2 and 3. The negative rating case shows the possibility of fraud. However if we ignore that, then rating users negatively is more apparent and visible than rating positively. This tells us that people prefer to criticize than to appreciate.
3. Since the number of edges with zero weight is zero, the dataset did not contain any instance of it. Also the dataset is a complete one, since there are no missing values or NAN or anything of that sort. A zero weight relation between two users simply signifies that either of them have not interacted with each other which is why neither have ever rated each other even once.

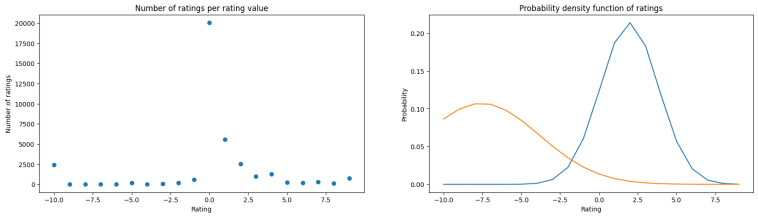

Figure 3 Number of rating of positive and negative vs their distributions

How can we improve?

Now we have got the basic analysis of data to make inference now. From the plots we can infer that positive weights although less (1,2,3) have a very high frequency. We can use this information to model the way new nodes in the form of edges are added to our model. Note that the experiments we conducted have weights normalized but here the weights are still between -10 and +10. It may serve as a unique insight for our experiments.

2.3 Implementation

We are examining how the introduction of new nodes affects a community in a graph. To do so, we must first establish communities within the graph. We employ correlation clustering to form these communities.

2.3.1 Correlation clustering

We use correlation clustering to group nodes based on the weights of their edges. These clusters are formed by connecting nodes through positive edges that meet the delta good condition. The delta good condition for a node is defined as follows:

$$N_v = \text{positive neighbours}(v) \tag{1}$$
$$\text{len}(N_v \cap C) \geq (1.0 - \delta) \cdot \text{len}(C) \text{ and } \text{len}(N_v \cap (\text{G nodes} - C)) \leq \delta \cdot \text{len}(C) \tag{2}$$

For a node to be in a cluster it should satisfy the delta good equation. Below is the pseudo code for the correlation clustering:

1. Check if there are any clusters available, if not, create an empty list.
2. While there are nodes that have not been clustered:
   a. Randomly sample all the nodes in the graph.
   b. Initialize a variable Av as None.
   c. For each vertex v in the sampled nodes:
      i. Get all positive neighbours of v.
      ii. For each neighbour x in the positive neighbours of v, remove x from Av if it does not satisfy delta good$(x, Av, 3 \times \text{delta})$.
      iii. Add all the nodes y that satisfy delta good$(y, Av, 7 \times \text{delta})$ to a set Y.
      iv. Update Av to be the union of Av and Y.
      v. If the size of Av is greater than 0, then a cluster has been found and exit the loop.
   d. If no cluster is found, then exit the loop.
   e. Append Av to the list of clusters.
   f. Remove all the nodes in Av from the set of nodes that have not been clustered.

2.3.2 Methodology

After performing clustering on the graph, we select the top ten nodes in each cluster based on their degree. Subsequently, we introduce new nodes to the graph that will be inserted into our network. In order to simulate a real-world scenario, we consider the possibility that a node already present in our graph with a higher degree will have a greater likelihood of interacting with the newly added nodes than a node with a lower degree. Thus, we prioritize nodes with a higher degree when selecting nodes to interact with the new nodes.

We choose two nodes from each cluster having at least 10 nodes with a probability function based on their degree, which will be connected to the newly introduced nodes in the graph. Note: We have introduced 10 new nodes and selected two nodes from each cluster having at least 10 nodes to connect to these newly added nodes.

We then define multiple trustworthiness metrics for a cluster. They are given in section 3.

We proceed by predicting the edges within a cluster that were not present in our graph, both before and after the introduction of new nodes. Our prediction method involves utilizing the notions of fairness and goodness[1] for the nodes, and we define the weights for these edges as follows:

$$weight(u, v) = fairness(u) \times goodness(v) \tag{3}$$

We proceed to evaluate the impact of adding new nodes to the graph on the weights prediction of the edges within a cluster. This allows us to quantify the changes that occur in the cluster upon addition of new nodes. Subsequently, we compare the effects on predicted weights after adding new nodes to the cluster's trustworthiness, which we previously evaluated. Note: The edges inside a cluster are those which have both the source and target present inside the cluster.

3 Trustworthines Metrics

We define various trustworthiness metrics for a cluster as follows:

$$metrics\_1[3] = \frac{(in\_p[i] + abs(out\_n[i]))}{(len(clusters[i]))} \; for \; cluster \; i \tag{4}$$

The metric described above calculates the sum of weights for positive edges entering a cluster and negative edges leaving the cluster. This metric is intuitively sensible because a cluster with higher trustworthiness is likely to have more positive edges entering it, while a higher number of negative edges leaving the cluster indicates that it rates other clusters negatively more frequently than other clusters. We have then averaged out this summation with the number of nodes present inside the cluster.

$$metrics\_2 = \frac{(in\_p[i] + abs(out\_n[i]))}{len(H.edges)} \; for \; cluster \; i \tag{5}$$

The above metric considers the sum of positive edges that are coming inside the cluster and absolute of the edges with negative weights going outside the cluster. The sum is then divided by the total number of edges in the whole graph. The basic intuition is here we re-scale the net weights coming in and out of a cluster by the same factor which is the number of edges in the whole graph.

$$metrics\_3 = \frac{(out\_p[i] + abs(in\_n[i]))}{(len(clusters[i]))} for \; cluster \; i \tag{6}$$

Here we consider the absolute sum of the weights going outside. We add the positive weights going outside the cluster and add the absolute value of the negative weights coming inside the cluster. Both of them act opposite to each other. The larger the value of out positive weights, the higher the trustworthiness of the clusters. The larger the value of in negative weights, the lower the trustworthiness of the cluster. We divide by the length of the cluster to remove the bias of large/small cluster edges sum.

$$metrics\_4 = \frac{(out\_p[i] + abs(in\_n[i]))}{(len(H.edges))} for \; cluster \; i \tag{7}$$

This metric can be seen as complementary to Metric 1, as it focuses on the outgoing positive edges and incoming negative edges. Unlike Metric 1, it does not directly measure trustworthiness; rather, it measures the level of distrust for a community. Specifically, a community with a higher number of incoming negative edges can be seen as more untrustworthy, as it receives more negative ratings from other communities.

4 Results and Conclusion

We conducted experiments as mentioned in the methodology to check the robustness of our proposed metrics. We used 12.5 % of the dataset for this purpose. To impart randomness, we did not set the random seed value. The different results generated by setting two different random seeds are given in experiment 1 and experment 2 respectively. The following tables show the difference in the weights observed in ascending order after introducing the new node in the two experiments.
To normalize for the difference in the number of missing edges within the cluster, we have computed the difference in the predicted weights for the missing edges and divided it by the number of missing edges within the cluster.

We have introduced only 10 new nodes in the existing graph owing to the computational constraints. Due to this the change in the predicted edges of the missing weights is low.
We evaluated the proposed metrics on the graph before introducing the new node to find the trustworthiness value corresponding to each cluster.

In all the plots below the y-axis represents the trustworthiness value and the x-axis represents the cluster number. The red highlighted points are corresponding to the clusters that have more than 10 nodes for which we have calculated the change of weights. The value mentioned corresponds to the change in weight corresponding to these clusters as mentioned in the table above.

Cluster Number	Difference in Weights
14	9.64×10^{-5}
8	1.00×10^{-4}
0	1.05×10^{-4}
2	1.10×10^{-4}
13	1.25×10^{-4}
12	1.30×10^{-4}
5	1.82×10^{-4}
28	2.32×10^{-4}
24	2.67×10^{-4}
23	2.80×10^{-4}
22	2.97×10^{-4}
19	3.80×10^{-4}
32	4.87×10^{-4}
30	6.72×10^{-4}
18	1.39×10^{-3}

(a) Experiment - 1

Cluster Number	Difference in Weights
12	1.01×10^{-4}
10	1.05×10^{-4}
0	1.18×10^{-4}
7	1.37×10^{-4}
2	1.46×10^{-4}
26	1.60×10^{-4}
15	2.39×10^{-4}
32	2.43×10^{-4}
4	2.57×10^{-4}
36	2.85×10^{-4}
13	3.05×10^{-4}
24	3.22×10^{-4}
18	3.34×10^{-4}
21	3.83×10^{-4}
28	4.74×10^{-4}
34	6.68×10^{-4}
20	1.30×10^{-3}

(b) Experiment - 2

Figure 4 Comparison of experiment results

4.1 Experiment - 1

As seen in Figure 5 for Metrics-1, the clusters with the least average change in the predicted weights post introduction of new nodes (i.e. clusters 14,8,0,2) occupy higher peaks. Thus, based on this metric, the more trustworthy cluster tends to have less change in the weights of the missing edges.

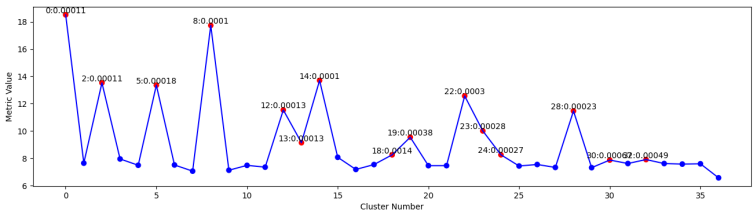

Figure 5 Metrics-1 value corresponding to the clusters for Experimentation - 1

As seen in Figure 6 for Metrics-2, with respect to the clusters having the least change in the predicted weights, there is no trend. Thus, this metric does not show any correlation between metric value and weight change.

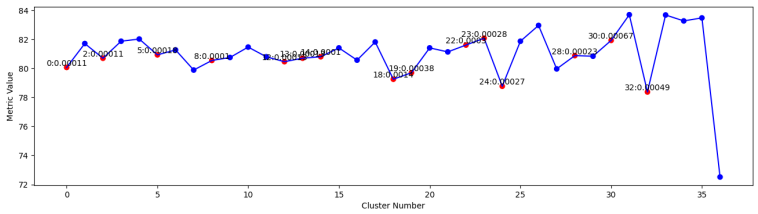

Figure 6 Metrics-2 value corresponding to the clusters for Experimentation - 1

As seen in Figure 7 for Metrics-3, the clusters with the least changes (i.e., clusters 0,8,14,2) occupy the highest peak in the plot. Thus, this metric clearly shows the correlation between the peaks and the average change in the weights of the predicted edges.

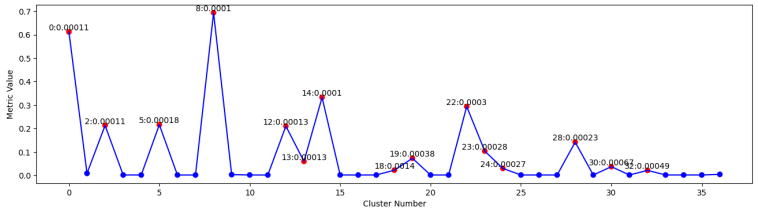

Figure 7 Metrics-3 value corresponding to the clusters for Experimentation - 1

As seen in Figure 8 for Metrics-4, the clusters with the least changes (i.e., clusters 0,8,14,2) occupy the lowest points in the plot. Thus, this metric clearly shows the correlation between the lowest points and the average change in the weights of the predicted edges.

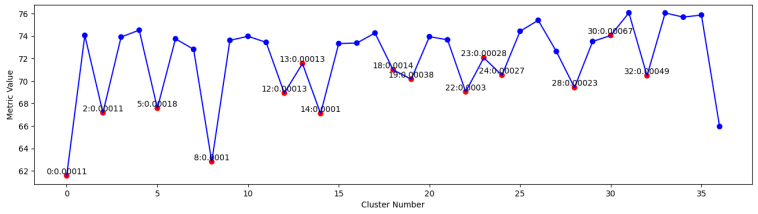

Figure 8 Metrics-4 value corresponding to the clusters for Experimentation - 1

4.2 Experiment - 2

As seen in Figure 9 for Metrics-1, the clusters with the least change in the weights are cluster number 12,10,0,7. Here, the cluster numbers 0 & 7 occupy higher peaks, whereas even though 12 & 10 occupy peaks they are not major peaks. Thus, based on Experiment - 1 and Experiment - 2, this metric provides insightful results but can be made more robust to get better outcomes.

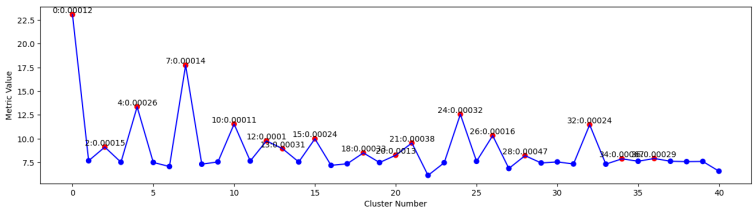

Figure 9 Metrics-1 value corresponding to the clusters for Experimentation - 2

As seen in Figure 10 for Metrics-2, the clusters with the least weight change show no distinctive behavior. Thus, based on Experiment-1 & Experiment-2, there are no insightful results that we can gain from this metric.

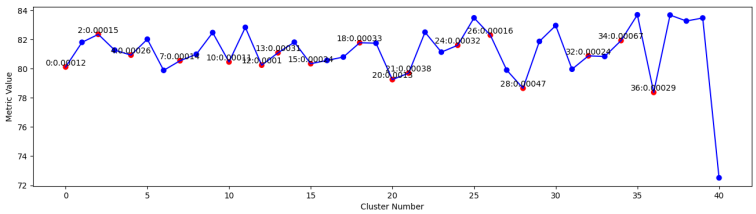

Figure 10 Metrics-2 value corresponding to the clusters for Experimentation - 2

As seen in Figure 11 for Metrics-3, all the clusters with the least change occupy peaks, although, except for clusters 0 & 7, others are not major. Thus, based on Experiment-1 & Experiment-2, this metric provides insightful results but can be made more robust for better outcomes.

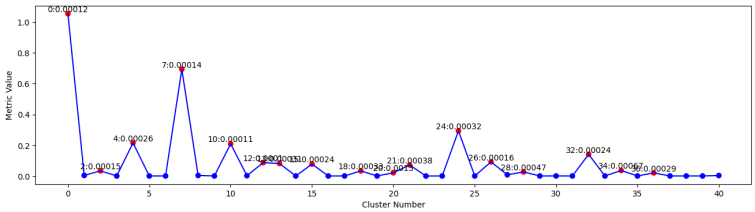

Figure 11 Metrics-3 value corresponding to the clusters for Experimentation - 2

As seen in Figure 12 for Metrics-4, all the clusters with the least change occupy valleys, and the clusters with the least change (i.e., cluster number 12,10,0,7) occupy the lowest point. Thus, based on Experiment-1 & Experiment-2, this metric provides insightful results.

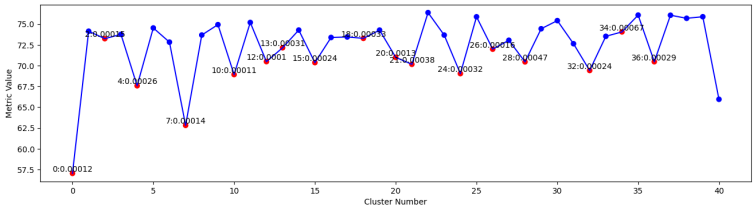

Figure 12 Metrics-4 value corresponding to the clusters for Experimentation - 2

4.2.1 Conclusion

Based on the results of the experiments it is evident that Metrics 1 & 3 clearly show that higher the value of the metric lower will be the change in the average predicted weights of the edges upon the introduction of the new node to the graph. Whereas for Metrics 4 lower the value lower the change in weights. Metrics 2 did not show any particular trend.

5 Future Work

- Currently, due to limited computation power, we have added only 10 nodes to the graph. Optimizing the code and using more computation power will help us draw better conclusions.
- We plan to make the metric more robust by capturing the complete trend for change in weight with the metric.

6 References

References

1 Srijan Kumar, Francesca Spezzano, VS Subrahmanian, and Christos Faloutsos. Edge weight prediction in weighted signed networks. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 221–230. IEEE, 2016.

2 Jure Leskovec and Rok Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1, 2016.

3 Harsh Patel, Shivam Sahni, and Pushkar Mujumdar. Trust as a metric for resiliency in signed social networks, 2021. `arXiv:2108.08665`.