Virtual Societies Project Research Paper

Prediction of Edge Weights and Community Detection in Weighted Signed Bitcoin Trust Networks

Filip Trajkovski – 171005

Faculty of Computer Science and Engineering, Skopje

August 2021

Abstract

Bitcoin has become an increasingly popular cryptocurrency because of the anonymity it offers. It was the first example of a valuable digital asset without any central authority like a bank. Any transaction between two bitcoin users is recorded in a public ledger called blockchain. We consider two marketplaces to exchange bitcoins and dollars, Bitcoin OTC and Bitcoin Alpha.

In this research paper, we will analyze these trust networks formed by users of Bitcoin as Weighted Signed Networks (WSNs). Since anonymity is the main feature in bitcoin transactions, a user's reputation needs to be qualified to avoid fraudulent transactions. The main area of research is the introduction to the problem of edge weight prediction in WSNs, as well as applying several models (in particular Fairness and Goodness, Reciprocal, Bias and Deserve, and Signed Hits) and comparing their performance. Furthermore, we will investigate if we can use different community detection algorithms to separate the fraudulent users from the honest ones.

I. Introduction

Bitcoin is the longest running and most well-known cryptocurrency, first released as open source in 2009 by the anonymous Satoshi Nakamoto. Introducing a radical new concept for money and currency, Bitcoin serves as a decentralized medium of digital exchange, with transactions verified and stored in a distributed public ledger without the need of a central intermediary. Transaction blocks contain an SHA-256 cryptographic hash of previous transaction blocks, and are therefore "chained" together, serving as an immutable record of all transactions that have ever occurred [3]. As might be expected, Bitcoin has been the biggest recent revolution in the financial space. The price of one bitcoin has increased from virtually nothing to an all-time high above $60,000 in April 2021 (see figure 1). Its introduction as a new programmable virtual money has led to the emergence of a whole cryptocurrency space, including Ethereum [4].

In the past several years, the ability to transfer unlimited amount of money without paying fees to a third party, and the idea of mining Bitcoin using the blockchain technology, caught the users' interest. Following this trend, a lot of platforms were created where users can make transactions based on certain market policy. When a user registers on such platform

and it has been authenticated, he/she can later use it to trade bitcoins. On some platforms, when a transaction has been completed, users can rate each other. The rating indicates how trustworthy the user is and helps other users to estimate this user whenever they want to make a transaction with him/her in the future.

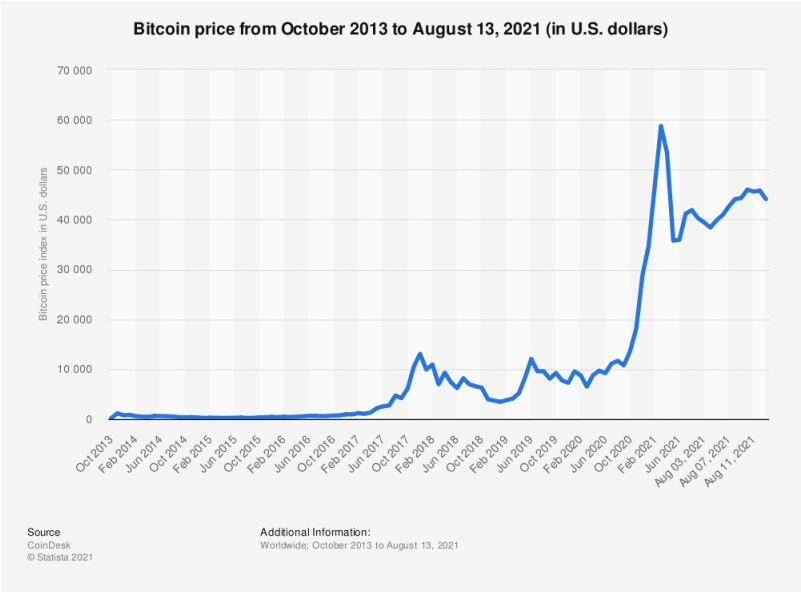

Figure 1: Bitcoin price from October 2013 to August 2021 in U.S. dollars

In this paper, we focus on two marketplaces to trade bitcoins against dollars – Bitcoin OTC and Bitcoin Alpha. The advantage of using these smaller networks is that each edge corresponds to some weight, the rating from user *u* to user *v*. This forms a web of trust between users, and it allows two users who don't know each other to perform a transaction based on the aggregated trust that they possess. These two networks are examples of Weighted Signed Networks (WSNs) where edges are labeled with positive and negative weights. WSNs can capture like/dislike, trust/distrust, agreement/disagreement, and other social relationships between people [1].

Our goal in this paper is to perform network analysis on these trust networks like Bitcoin OTC and Bitcoin Alpha and apply various algorithms in the field of edge weight prediction and community detection. We base a good amount of our analysis on [1], which defines fairness and goodness as two intertwined measures that describe each node in a WSN.

II. Related work

There is substantial work on predicting edge weights in unsigned social networks. This work includes baselines such as reciprocal edge weight [10], and triadic balance and status measures [11, 12]. Two popular unsupervised algorithms are EigenTrust [6] and TidalTrust [7]. EigenTrust calculates a global value of trust for each vertex by finding the left eigenvector of a normalized trust matrix. TidalTrust calculates the trust from a source to a sink, by recursively propagating trust via the sink's predecessors till it reaches source. However, these papers deal with edge weight prediction in unsigned social networks, while WSN is a new problem deriving from SSN.

Leskovec et al. [5] use balance and status theories to predict edge signs (a measure of trust) in a directed network. Kumar et al. [1] proposed the metrics, goodness and fairness, to compute the edge weight between two nodes (trust between two anonymous users). The goodness of a node intuitively captures how much this node is liked/trusted by other nodes, while the fairness of a node captures how fair the node is in rating other nodes' likeability or trust level. When compared against several individual algorithms from both the signed and unsigned social network literature, the fairness and goodness metrics almost always have the best predictive power. This is the first paper to show how to predict edge weights in weighted signed networks.

In [2], the authors present REV2, a system to identify fraudulent users on rating platforms such as Bitcoin trading networks. Three interdependent intrinsic quality metrics have been proposed – fairness of a user, reliability of a rating and goodness of a product. The fairness and reliability quantify the trustworthiness of a user and rating, respectively, and goodness quantifies the quality of a product. REV2 algorithm is designed to calculate these intrinsic scores for all users, ratings, and products by combining network and behavior properties.

III. Datasets and visualizations

On Bitcoin OTC, people can exchange bitcoins and build up trust as they take part in more exchanges. Users rate each other in a scale of -10 (total distrust) to 10 (total trust). The guidelines for the rating are available on the wiki page for OTC and are shown in Table 1. We can see that because there are only recommendations for ratings -10, -1, 1, 5, 8 and 10, we have a higher amount of ratings for these values. The most common rating is 1, which corresponds to the first exchanges between two users that go well. 6.78% of ratings are -10 which means that there is a significant number of fraudulent transactions where one user never sends the money. The dataset is available on SNAP[1] and is the first public WSN.

Rating	Fraction	Guideline
10	2.1%	You trust this person as you trust yourself.
9	0.3%	
8	0.78%	Large number of high-value transactions, long period of association, very trustworthy.
6, 7	1.3%	
5	3.6%	You've had a number of good transactions with this person.
2, 3, 4	25.5%	
1	56.3%	One or two good transactions with this person.
-1	1.7%	Person strikes you as bit flaky. Unreasonable/unexpected delays in payment, etc.
-2 to -9	1.5%	
-10	6.78%	Person failed to hold up his end of the bargain, took payment and ran, fraudster.

Table 1: Rating guidelines from the OTC wiki and fraction of the overall ratings for each rating.

[1] https://snap.stanford.edu/data/soc-sign-bitcoin-otc.html

The Bitcoin Alpha Trust network dataset, which is also available on SNAP[2], is similar in almost every way to the Bitcoin OTC trust network. It is also a weighted directed graph and has ratings from -10 to 10.

As can be easily seen from the provided information on SNAP, Bitcoin Alpha has 3,783 nodes and 24,186 edges, while Bitcoin OTC has 5,881 nodes and 35,592 edges.

First, we created directed weighted graphs using the NetworkX library where the weights represent the trust between users (i.e. the rating they gave one another). Then, we made visualizations of a subsample of both graphs which included ratings from one month only. A suitable option for this was March 2013 because it has a good spread of ratings. For layout of the nodes and their positioning in the graph, we used the Fruchterman-Reingold force-directed algorithm [13] which enforces several rules:

- Larger nodes have more reviews/ratings
- Green nodes are positively reviewed/rated on average
- Red nodes are negatively reviewed/rated on average
- Color intensity is degree of trust (positive or negative)
- Edges are directed
- Longer edges are more negative, shorter edges are more positive

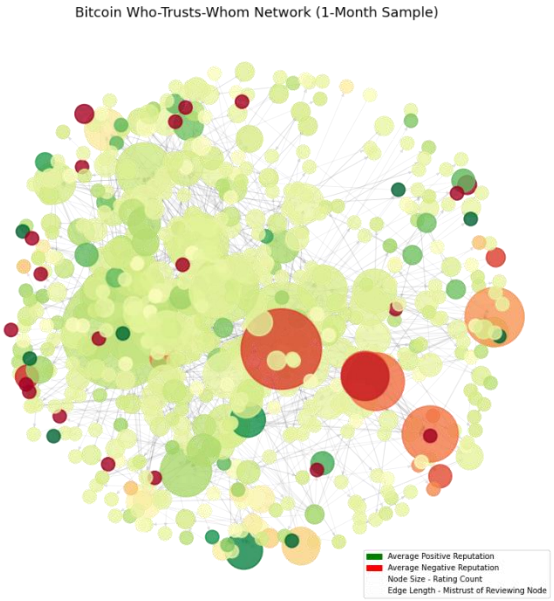

Figure 2: Bitcoin OTC Who-Trusts-Whom Network

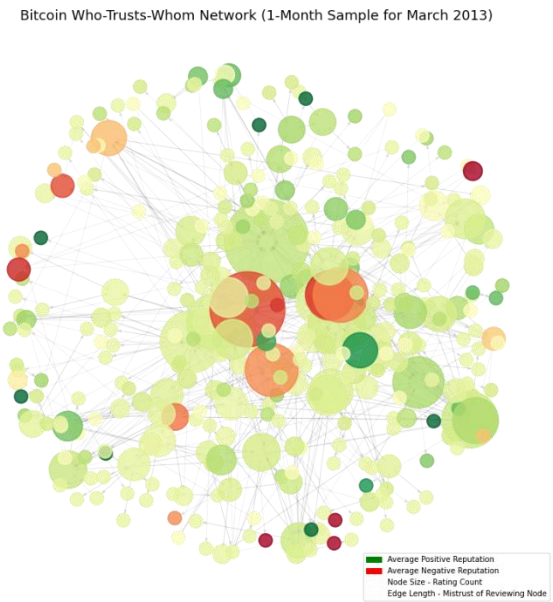

Figure 3: Bitcoin Alpha Who-Trusts-Whom Network

We expect trusted nodes to transact more with each other, appear greener and larger, and have on average more positive reviews, and thus shorter edges. These should cluster together. Nodes with lower trust should appear smaller and have longer edges. With the Fruchterman-Reingold algorithm, they should be pushed away from trusted clusters. On figures 2 and 3, the visualizations with the force-directed algorithm for Bitcoin OTC and Bitcoin Alpha graphs are shown respectively. We can observe that most of our expectations are confirmed as highly negatively rated nodes (dark red) appear to be pushed to the

[2] https://snap.stanford.edu/data/soc-sign-bitcoin-alpha.html

perimeter and away from positive (green) clusters. However, we can notice there are some larger red nodes that are well integrated and infiltrated in the network by being positioned in the center.

In the dataset, the users were not labeled as honest or fraudulent, so we don't have ground truth and must classify them ourselves. For this part, we used a simple calculation that considers every user as fair, so we just summed the weights of every incoming edge for each node and based on its value we classified the particular user as fraudulent or honest. If the sum is positive that means the user is seen as good by others, otherwise we consider him/her as dishonest. The honest users in Bitcoin OTC and Bitcoin Alpha networks are colored blue, while the fraudulent ones are represented with red color on figures 4 and 5, respectively.

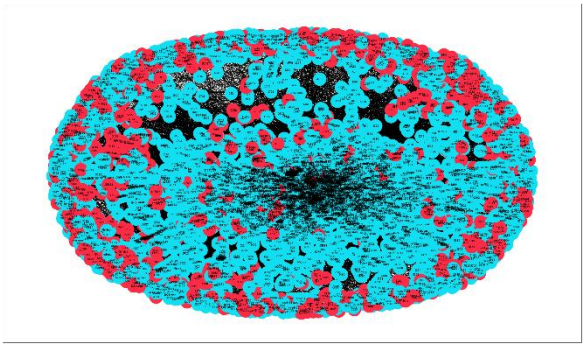

Figure 4: Visualization of users in Bitcoin OTC

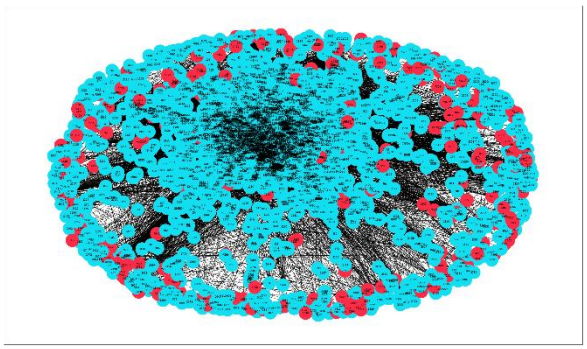

Figure 5: Visualization of users in Bitcoin Alpha

IV. Methods for predicting edge weight

A Weighted Signed Network (WSN) is a directed, weighted, graph $G = (V, E, W)$ where *V* is a set of users, $E \subseteq V\ x\ V$ is a set of edges, and $W : E -> [-1, +1]$ is a mapping that assigns a value between -1 and +1 to each edge. $W(u, v)$ can be thought of as assigning a degree of "likes", "agrees" or "trust" score describing how much user *u* likes a user *v*.

As part of the research, we have investigated a common set of algorithms associated with edge prediction and community detection in social networks.

1. Fairness and Goodness

The fairness and goodness model introduced in [1] is the state-of-the-art algorithm for edge weight prediction in weighted signed networks. These metrics are based on the intuition that a 'fair' or 'reliable' rater should give a user the rating it deserves, while an 'unfair' one would deviate from that value. Therefore, the ratings given by unfair users should be given low importance, while ratings given by fair users should be considered important. On the other hand, the goodness of a node specifies how much other nodes like/dislike, agree/disagree, or trust/distrust that node and what its true quality is. This is the rating a totally fair vertex would give. Higher goodness implies the vertex is more trustworthy in the network. Hence, a 'good' or 'trustworthy' node would receive many high positive ratings

from fair nodes, while a 'non-trustworthy' node would receive high negative ratings from fair nodes.

To obtain the fairness and goodness of every node, we need to run an algorithm that will make multiple iterations until the values converge for each node.

Fairness and goodness need to satisfy two equations:

$$g(v) = \frac{1}{|in(v)|} \sum_{u \in in(v)} f(u) w(u, v)$$

$$f(u) = 1 - \frac{1}{|out(u)|} \sum_{v \in out(u)} \frac{|w(u, v) - g(v)|}{2}$$

The first equation means that a node is as good as the mean of its ratings, weighted by fairness. The second equation creates a fairness score in [0, 1] for *u* given how well the ratings of *u* correlate with true goodness.

By iterating on values of *f* and *g* using the equations above, we can converge to the true fairness and goodness for every node.

Input: A WSN $G = (V, E, W)$
Output: Fairness and Goodness scores for all vertices in V
Let $f^0(u) = 1$ and $g^0(u) = 1, \forall u \in V$
$t = -1$
do
  $t = t + 1$
  $g^{t+1}(v) = \frac{1}{|in(v)|} \sum_{u \in in(v)} f^t(u) \times W(u, v), \forall v \in V$
  $f^{t+1}(u) = 1 - \frac{1}{2|out(u)|} \sum_{v \in out(u)} |W(u, v) - g^{t+1}(v)|, \forall u \in V$
while $\sum_{u \in V} |f^{t+1}(u) - f^t(u)| > \epsilon$ or $\sum_{u \in V} |g^{t+1}(u) - g^t(u)| > \epsilon$
Return $f^{t+1}(u)$ and $g^{t+1}(u), \forall u \in V$

One of the direct applications of the fairness and goodness algorithm is to predict the weight of a missing edge. Given two nodes *u* and *v*, we can predict that the weight of the directed edge *(u, v)* will be $f(u) \times g(v)$. The intuition behind this predicted score is that *g(v)* represents how good node *v* is, and *f(u)* represents how fair *u* is. If *u* is totally fair, then the score of *(u, v)* should be the goodness of *v*, *g(v)*. However, if *u* is unfair, then we cannot really predict which value it will give to the weight, so we predict a value closer to zero by adding the factor *f(u)*. As in [1], we call this predicted score the *F x G* score of the edge.

Figures 6 and 7 show the fairness and goodness distribution for all nodes in the Bitcoin OTC and Bitcoin Alpha networks, respectively. It can be noticed that most vertices in both networks have very high fairness score meaning that most of the users are fair, but a small portion are not. When it comes to goodness, most nodes have low positive scores (between 0 and 1), while a considerable fraction is considered "not good" (around 14% have a negative score).

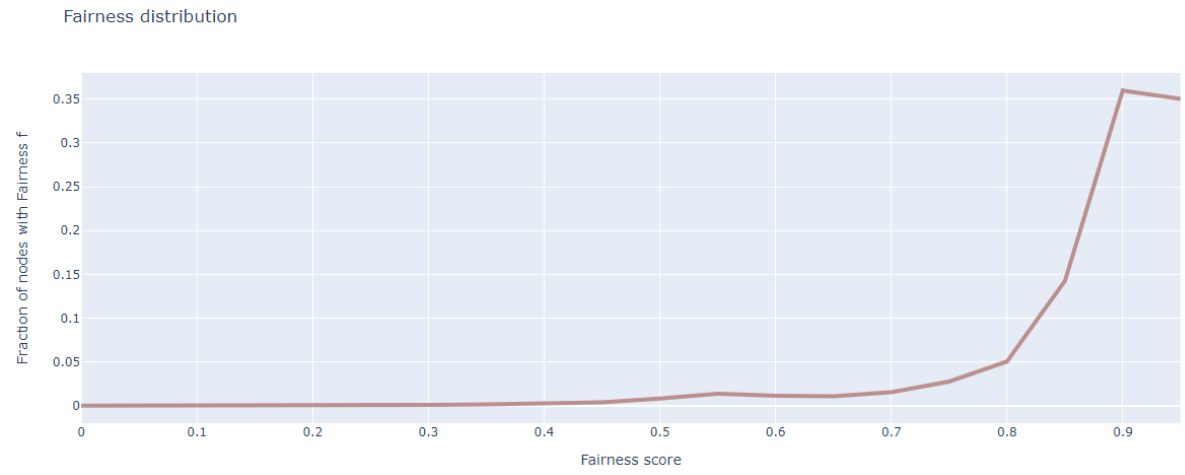

Figure 6: Fairness and Goodness distribution of Bitcoin OTC network

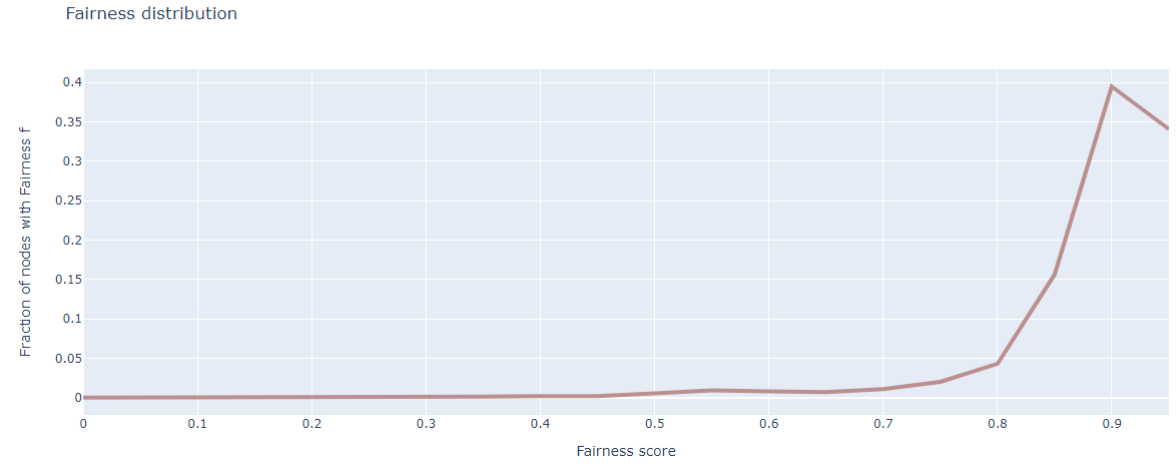

Figure 7: Fairness and Goodness distribution of Bitcoin Alpha network

2. Reciprocal

As a baseline of weight prediction, the reciprocal algorithm takes the weight of the directed edge *(v, u)* to be equal to the weight of *(u, v)*. When the reciprocal edge doesn't exist, the weight of that edge is set to 0. Therefore, the weight of *(u, v)* is predicted by:

$$W(u,v) = \begin{cases} W(v,u), & \text{if } u \to v \text{ exist} \\ 0, & \text{otherwise} \end{cases}$$

3. Bias and Deserve

Bias and Deserve method is proposed by Mishra and Bhattacharya in [8]. To compute "bias" and "deserve", we should first normalize the weights, and keep them in range of [-1, 1] where 0 is neutral opinion. Then we say node *u* gives a trust-score of w_{ij} to node *v* for a given rating *(u, v)*. The two attributes of a node are defined by:

- *Bias* which reflects the expected weight of an outgoing edge
- *Deserve* which reflects the expected weight of an incoming edge from an unbiased vertex

Let $d^o(u)$ denote the set of all outgoing edges from vertex *u* and likewise, $d^i(u)$ denote the set of all incoming links to node *u*. Then, bias and deserve are iteratively computed as:

$$BIAS^{(t+1)}(u) = \frac{1}{2|d^o(u)|} \sum_{v \in d^o(u)} [W(u,v) - DES^t(v)]$$

$$DES^{(t+1)}(u) = \frac{1}{2|d^i(u)|} \sum_{v \in d^i(u)} [W(v,u)(1 - X^t(v,u))]$$

where $X^t(v,u) = max\{0, BIAS^t(v) \times W(v,u)\}$

The iterative formulations of bias and deserve allow us to predict the weight of *(u, v)* based on the deserve *DES(v)* of vertex *v*. Thus, the weight is directly predicted by:

$$W(u,v) = DES(v)$$

4. Signed-Hits

Signed-Hits prediction is computed by using a modified version of HITS for signed network [9]. Signed-Hits will compute the hub and authority scores of every node separately on positive graph (all edges are positive) and negative graph, using the equations:

$$h^+(u) = \sum_{v \in out^+(v)} a^+(v); a^+(u) = \sum_{v \in^+(u)} h^+(v)$$
$$h^-(u) = \sum_{v \in out^-(v)} a^-(v); a^-(u) = \sum_{v \in^-(u)} h^-(v)$$

After convergence, we assign the authority score $a(u) = a^+(u) - a^-(u)$ and hub score $h(u) = h^+(u) - h^-(u)$ to each vertex u. Authority scores estimate the node value based on the incoming links, while hub scores estimate the node value based on outgoing links. We use weighted average to compute the weight prediction:

$$W(u,v) = \frac{h(u) * \sum_{z \in out(u)} W(u,z) + a(v) * \sum_{z \in in(v)} W(z,v)}{h(u) * |V_{z \in out(u)}| + a(v) * |V_{z \in in(v)}|}$$

In this equation, we use *h(u)* to be the weight of all out-going edges and *a(v)* to be the weight of all in-coming edges. The reason is that *h(u)* represents the node value based on outgoing edges. So, the predicted weight of out-edges from *u* should be higher if *h(u)* is high. Likewise, the predicted weight of in-edges into *v* should be higher if *a(v)* is high.

V. Methods for community detection

One of the common tasks when working with social networks is finding groups of densely connected nodes, also known as community detection. Because we are working with signed networks, we decided to remove the edges with negative weights and weights equal to zero in order to avoid grouping nodes that don't trust each other.

1. Label Propagation

Label Propagation is a semi-supervised machine learning algorithm that assigns labels to previously unlabeled data points [14]. This algorithm is probabilistic and the found communities may vary on different executions. After initializing each node with a unique label, the algorithm repeatedly sets the label of a node to be the label that appears most frequently among that node's neighbors. Ties are broken randomly and the order in which the vertices are updated is randomized before every iteration. The algorithm halts when each node has the label that appears most frequently among its neighbors.

2. K-Clique

This algorithm finds k-clique communities in a graph using the percolation method. A k-clique community is the union of all cliques of size k that can be reached through adjacent (sharing k-1 nodes) k-cliques [15].

3. Fluid Communities

The algorithm is based on the simple idea of fluids interacting in a non-homogeneous environment, expanding and contracting based on their interaction and density. Its initialization is random, so found communities may vary on different executions. First each of the initial k communities is initialized in a random vertex in the graph. Then, the algorithm iterates over all vertices in a random order, updating the community of each vertex based on its own community and the communities of its neighbors. This process is performed several times until convergence. At all times each community has a total density of 1, which is equally distributed among the vertices it contains. If a vertex changes a community, vertex densities of affected communities are adjusted immediately. When a complete iteration over all vertices is done, such that no vertex changes the community it belongs to, the algorithm has converged [16].

4. Leiden Algorithm

Leiden algorithm is similar to the famous Louvain algorithm but is faster and yields higher quality solutions [17]. It can optimize both modularity and the Constant Potts Model, which does not suffer from the resolution-limit. The Leiden algorithm consists of three phases:

- Local moving of nodes
- Refinement of the partition
- Aggregation of the network based on the refined partition, using the non-refined partition to create an initial partition for the aggregate network

In the local move procedure in Leiden algorithm, only nodes whose neighborhood has changed are visited. The refinement is done by restarting from a singleton partition within each cluster and gradually merging the subclusters. When aggregating, a single cluster may then be represented by several nodes (which are the subclusters identified in the refinement).

The Leiden algorithm is typically iterated: the output of one iteration is used as the input for the next iteration. At each iteration all clusters are guaranteed to be connected and well-separated. After an iteration in which nothing has changed, all nodes and some parts are guaranteed to be locally optimally assigned.

VI. Results

We conduct a first experiment following [1] by leaving one edge out of the graph and trying to predict its weight. We report two metrics – the root mean square error (RMSE), which is the square root of the mean square error between the predictions and the true weights and the Pearson Correlation Coefficient (PCC). The results for the four applied algorithms for

edge weight prediction in Bitcoin OTC and Bitcoin Alpha weighted signed networks are shown in tables 2 and 3, respectively.

Algorithm	RMSE	PCC
Fairness and Goodness	0.33	0.389
Reciprocal	0.316	0.511
Bias and Deserve	0.339	0.37
Signed Hits	**0.3**	**0.542**

Table 2: Performance of different algorithms for edge weight prediction in leave-one-out experiment for Bitcoin OTC dataset

Algorithm	RMSE	PCC
Fairness and Goodness	0.295	0.268
Reciprocal	**0.263**	**0.52**
Bias and Deserve	0.3	0.262
Signed Hits	0.288	0.254

Table 3: Performance of different algorithms for edge weight prediction in leave-one-out experiment for Bitcoin Alpha dataset

First, we were interested in testing whether users are impacted by how other people rate them. It can be observed that the prediction from the Reciprocal method outperforms Fairness and Goodness (*F x G*) model by a very small margin. The *F x G* method is agnostic to the social interactions between users *u* and *v*, and only considers the characteristics of nodes *u* and *v*. The reciprocal method, on the contrary, only uses social clues and retaliate on a rating by giving the same one. Furthermore, we noticed that a majority of edges are reciprocated in the network. As Bitcoin OTC and Alpha are networks of trust between users exchanging bitcoins, it seems logical that most users rate each other after a transaction, so most edges are reciprocated. Finally, we can conclude that among the algorithms we experimented with, the reciprocal method, though simple and naïve, had best performance. It has the best RMSE and PCC values for the Bitcoin Alpha network and second-best for the Bitcoin OTC network. This proves our hypothesis that raters are heavily influenced by how other people rate them and in the nature of the human behavior is to apply the talion law: “an eye for an eye”. Also, we can notice that the modified HITS algorithm for signed network provided good results in this task.

When it comes to community detection in the networks, the best results were achieved with the Label Propagation algorithm in both cases. For Bitcoin OTC, 136 communities were detected, most of which, precisely 115 (84.56%), contained nodes of only one type (honest or fraudulent). Similar results were achieved for the Bitcoin Alpha network where we found 97 communities with Label Propagation and 82 (84.54%) consisted of users from the same class. However, the unequal number of honest and fraudulent users presents a problem because we often detect one larger community which contains majority of the nodes. Therefore, most of the users in it would be honest and at the same time we will predict a good portion of fraudulent users from that community as honest. Another algorithm that has good performance is K-Clique which detected 137 communities in total for the Bitcoin OTC graph and 74 of them contained users of only one type. Better performance was

achieved for the Bitcoin Alpha network where 72 communities were detected and 57 of them consisted of just honest or fraudulent users.

VII. Conclusion

Based on the described results, we can conclude that what matters the most in a trust network is the user-to-user interaction. If user *u* rates *v*, the rating that *v* gave to *u* is more important than the overall goodness of *v*. In the context of marketplaces based on trust, this can be explained by the fact that both ratings depend on an underlying event: the actual trade of bitcoins and dollars.

Our second finding is that the Label Propagation algorithm is a simple way to separate honest from fraudulent users with significant accuracy. However, in the future other community detection algorithms should be investigated which might provide better performance than those proposed and applied in this research paper.

All visualizations, project code and results have been made available at https://github.com/trajkovskifilip/Bitcoin-Trust-Network-Analysis .

References

[1] S. Kumar, F. Spezzano, V. S. Subrahmanian, and C. Faloutsos. Edge weight prediction in weighted signed networks. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 221-230. *IEEE*, 2016.

[2] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. S. Subrahmanian. REV2: Fraudulent user prediction in rating platforms. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 333-341. *ACM*, 2018.

[3] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Cryptography Mailing List*. 2009.

[4] G. Wood. Ethereum: A secure decentralized generalized transaction ledger. *Ethereum project yellow paper*, vol. 151, pages 1-32. 2014.

[5] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In *Proceedings of the Conference on Human Factors in Computing Systems*. *ACM*, 2014.

[6] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th International Conference on World Wide Web,* pages 640-651. *ACM*, 2003.

[7] Y. Katz, and J. Golbeck. Social network-based trust in prioritized default logic. *Journal of Web Semantics*. 2006.

[8] A. Mishra, and A. Bhattacharya. Finding the bias and prestige of nodes in networks based on trust scores. In *Proceddings of the 20th International Conference on World Wide Web*, pages 567-576. *ACM*, 2011.

[9] M. Shahriari, and M. Jalili. Ranking nodes in signed social networks. *Social Network Analysis and Mining*, vol. 4, no. 1. 2014.

[10] E. Gilbert. Predicting tie strength in a new medium. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, pages 1047-1056. *ACM*, 2012.

[11] E. Gilbert, and K. Karahalios. Predicting tie strength with social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 211-220. *ACM*, 2009.

[12] S. Sintos, and P. Tsapras. Using strong triadic closure to characterize ties in social networks. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1466-1475. *ACM*, 2014.

[13] T. M. J. Fruchterman, and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11). 1991.

[14] G. Cordasco, and L. Gargano. Community detection via semi-synchronous label propagation algorithms. In *Business Applications of Social Network Analysis (BASNA), 2010 IEEE International Workshop on*, pages 1-8. IEEE, 2010.

[15] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, no. 435, pages 814-818. 2005.

[16] F. Pares, D. Garcia-Gasulla et al. Fluid communities: a competitive, scalable and diverse community detection algorithm. In *Proceedings of the 6th International Conference on Complex Networks and Their Applications*, pages 229-240. *Springer*, 2017.

[17] V. A. Traag, L. Waltman, and N. J. Van Eck. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1), pages 1-12. 2019.