CS 224W Project Report
Network Analysis of Weighted Signed Bitcoin Networks

Akshay Kumar, Vincent Ying

December 11, 2017

Abstract

Bitcoin has become an increasingly popular cryptocurrency because of the anonymity it offers. Any transaction between two bitcoin users is stored in a public ledger (called blockchain). While there has been analysis of various Bitcoin networks [RS12, FKP14, OKH13], most of the papers consider only unsigned Bitcoin networks and only do basic analysis of the transaction graph, such as amount of bitcoin exchanged. In this paper, we will analyze the trust network formed by users of Bitcoin as weighted signed networks. Since anonymity is the main feature in bitcoin transactions, a user's reputation needs to be qualified to avoid fraudulent transactions. Towards this goal, we will study signed, weighted bitcoin trust networks. Network properties of these networks will be examined and the theories of balance and status from the social-psychological domain will be applied to model user trust.

1 Introduction

Bitcoin is a peer to peer cryptocurrency system which allows users to transact cryptocurrency electronically and anonymously. To maintain anonymity and ensure that a user is not defrauded during a transaction, users neeed a measure of trustworthiness and to validate other users reputation. High reputation can be quantified through a trust network of users, where a user can assign a trust level (rating) to other users in the network. In this paper, we will perform analysis through the social-psychological point of view from the theories of balance and status.

2 Prior Work

Ever since the inception of Bitcoin with the whitepaper by Nakamoto [Nak08], there has been much interest and research in how Bitcoin functions and its network properties. How transactions are carried out under anonymity for both parties and the de-anonymization of bitcoin users have been of particular interest.

Fleder et al. [FKP14] explores the amount of anonymity in the Bitcoin system by scrapping emails and forum posts for valid Bitcoin addresses and matching it to an exact Bitcoin transaction. Ron et al. [RS12] perform a quantitative analysis of the full Bitcoin transaction graph and partially deanonymize the Bitcoin network based on time and volume of transactions. Decker et al. [DW13] investigate how bitcoin propagates information in the network and the growth of the blockchain.

Leskovec et al. [LHK14] use balance and status theories to predict edge signs (a measure of trust) in a directed network. Kumar et al. [KSSF16] proposed the metrics, *goodness* and *fairness*, to compute the edge weight between two nodes (trust between two anonymous users).

Most of the papers focusing on Bitcoin networks examines the transaction graph and describes some structural property of the graph. There has not been as much research on the trust network for the Bitcoin graph ([KSSF16]). However, Tang et al. [TCAL16] provides a good summary of signed network analysis.

3 Data Collection

We will perform our analysis on two main data sets: **bitcoin-alpha** and **bitcoin-otc**. Both bitcoin-alpha and bitcoin-otc are trust networks comprised of user ratings on various IRC channels. A user can rate another user between -10 (extremely distrustful) to +10 (totally trustful) in increments of +1 units. Both the networks are directed and weighted. Bitcoin-alpha has 3,783 nodes and 24,186 edges. Bitcoin-otc has 5,881 nodes and 35,592 edges.

4 Initial findings and mathematical background

Let the bitcoin trust graph $G = G(N, E, w)$ have vertex set N, edge set E, and the weight function $w : E \to [-10, 10]$.

For every edge are three factors to consider: edge direction (directed or undirected), sign, and weight. We will dissect the graph metrics by all three types.

4.1 Degree and Weight Distribution

We generalize the notion of node degree to the weight of a node. The incoming weight of a node is the sum of weights of all incoming edges to that node. Similarly, the outgoing weight of a node is the sum of weights of all outgoing edges from that node.

For directed graphs, the plots for the various weight distributions are below.

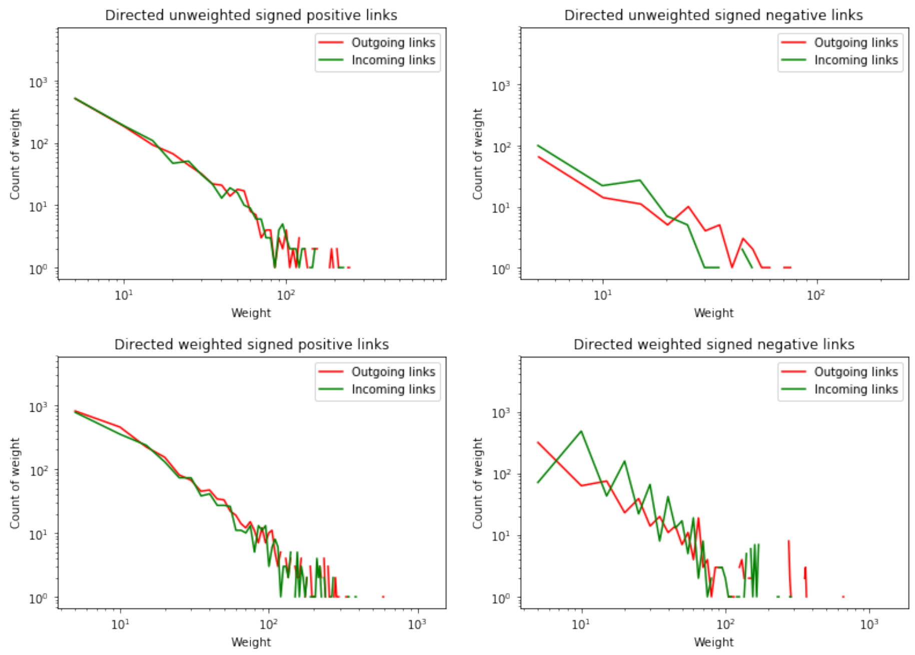

Figure 1: Degree/weight distribution

The graphs above show that negative links do not strictly follow power law. The power law exponent is close to 2.3 for all cases, except negative incoming links. For those links, the power law exponent is close to 3.9. This is most likely caused by noise or some other distortion in data. The results here are for the Bitcoin-otc network but the Bitcoin-alpha network also has similar results.

4.2 Clustering coefficient

Next, we examined the average clustering coefficient of the Bitcoin graphs. For an undirected and unweighted graph, the clustering coefficient of a given node is the ratio between number of edges to the neighbors and maximum number of edges that can potentially exist between the neighbors. If degree of a node v is k, then the clustering coefficient is

$$C = \frac{2|\{e_{jk} : v_j, v_k \in N, e_{jk} \in E\}|}{k(k-1)}$$

The average clustering coefficient is simply the average of all nodes' clustering coefficients.

We extend this definition to directed graphs as well. If node a has outgoing edges to nodes $\{b, c\}$ and nodes $\{b, c\}$ have an edge, a cluster is formed (as shown in Figure 2). The clustering coefficient can then be formulated as

$$C = \frac{|\{e_{jk} : v_j, v_k \in N, e_{jk} \in E\}|}{k(k-1)}$$

If k_i is the degree of node, the maximum possible number of edges between its neighbors is $k(k-1)$, since we also need to consider edge direction.

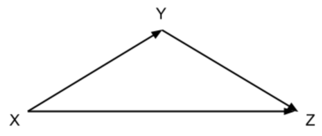

Figure 2: Directed cluster triangle

To extend the clustering coefficient calculation to unweighted, signed networks, we consider the below four triads along with their signs under balance theory. The approach we have taken follows the one suggested in [KLB09].

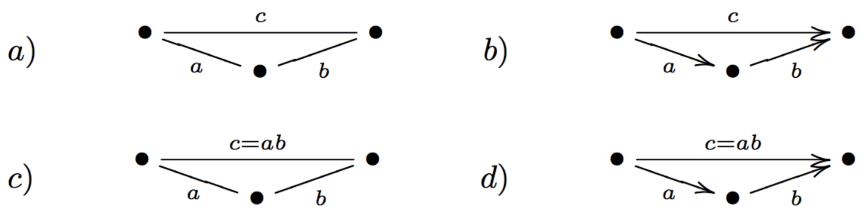

Figure 3: Clustering triad possibilities.[KLB09]

In Figure 3, the labels a, b and c denote the edge signs. If we take into account edge sign while computing clustering coefficients, the cluster is valid only if the edge signs match, $i.e$ $c = a \times b$.

Then, we extended calculation of the clustering coefficient to weighted (unsigned) networks. Define w as the signed weight function. Temporarily disregard edge sign. Let a be the unweighted analogue of w. In terms of an adjacency matrix, one way to think about the clustering coefficient for a vertex x is as follow [IE07]:

$$C(x) = \frac{\sum_{\substack{y,z \in \Pi(x) \\ y \neq z}} a(x, y)a(y, z)a(x, z)}{\sum_{\substack{y,z \in \Pi(x) \\ y \neq z}} a(x, y)a(x, z)}$$

Here, $\Pi(x)$ denotes the set of outgoing neighbors of x. Next, generalize weighted networks by replacing a with w:

$$C(x) = \frac{\sum_{\substack{y,z \in \Pi(x) \\ y \neq z}} w(x, y)w(y, z)w(x, z)}{\sum_{\substack{y,z \in \Pi(x) \\ y \neq z}} w(x, y)w(x, z)} \tag{1}$$

To compare average clustering coefficient of weighted networks versus unweighted networks, we normalize weights by dividing them by $\max_{x,y} w(x, y)$.

Finally, we combine the previous two approaches to compute the average clustering coefficient for signed weighted networks. The weights utilized are described in Equation 1, but a triad is valid only if it satisfies the sign requirements in Figure 3.

The average clustering coefficient for the different cases are tabulated below.

Directed	Signed	Weighted	Average CC
N	N	N	0.29
Y	N	N	0.23
N	N	Y	0.12
Y	N	Y	0.08
N	Y	N	0.26
Y	Y	N	0.22
N	Y	Y	0.10
Y	Y	Y	0.08

Directed	Signed	Weighted	Average CC
N	N	N	0.28
Y	N	N	0.23
N	N	Y	0.11
Y	N	Y	0.08
N	Y	N	0.24
Y	Y	N	0.21
N	Y	Y	0.09
Y	Y	Y	0.08

Table 1: bitcoin-otc (on the left) and bitcoin-alpha (on the right)

Clustering Coefficient Observations:

1. Average clustering coefficient of Bitcoin-alpha is less than that of Bitcoin-otc. This is to be expected since Bitcoin-alpha has fewer edges.
2. Average clustering coefficient of directed networks is less than undirected networks since edge directions must match.
3. Average clustering coefficient of signed networks is less than that of unsigned networks because edge signs must match.
4. Average clustering coefficient of weighted networks is less than that of unweighted networks.

The last three points apply in combination with each other.

4.3 Balance Theory

Besides the general properties of the two Bitcoin networks, we have also collected statistics based on the Balance theory. This model originates from social psychology and the modeling of real-world relationships between three individuals that know each other. Their relationships are labeled either positive or negative (like or dislike). A triad is balanced if there is an odd number of positive edges. A triad with one or three positive edges is stable. If a triad has one or three negative edges, it is unbalanced. Even though the model was meant for undirected networks, we applied it to directed networks by disregarding the direction of the edges [LHK14]. The four triad possibilities are given below.

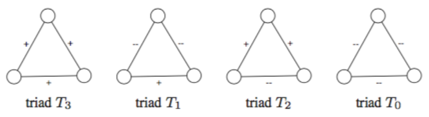

Figure 4: Triad possibilities for two given signed edges. [LHK14]

For every triad in the network, we counted the number of triads for each of the four types. For a triad in the network, the probability of it belonging to any given type is detailed in the table below.

Balance Triad Statistics

		Balanced		Unbalanced	
Graph	$\|T_i\|$	$p(T_3)$	$p(T_1)$	$p(T_2)$	$p(T_0)$
bitcoin-alpha	22153	0.7600	0.1466	0.0859	0.0075
bitcoin-otc	33493	0.6873	0.1324	0.1699	0.0104

Triads T_0 and T_2 are unbalanced since there are an odd number of edges. The overall stability of a network can be inferred by the number of triads of type T_0 and T_2 present. The probability that a given triad is unbalanced can be found by adding $p(T_0) + p(T_2)$.

One weakness of this definition is that these relationships may be too strict. The degree of imbalance in triad $T2$ can be argued to be greater than that present in T_0. A weaker version of balance is then defined by allowing triads with all negative edges [CHN$^+$14].

4.4 Status Theory

Another way to model trust between nodes in a network is Status theory. Status theory applies the concept of social status to networks of signed links. The three nodes in a given triad are assigned three different levels of status, given the arrangement of the signed edges and their direction. The predictions made by balance theory (which is based on undirected networks) and status theory often differ.

The count of of sixteen different triads detailed by status theory were collected for the two Bitcoin networks. As can be seen in the table below, triad t_1 is overwhelmingly the most prevalent.

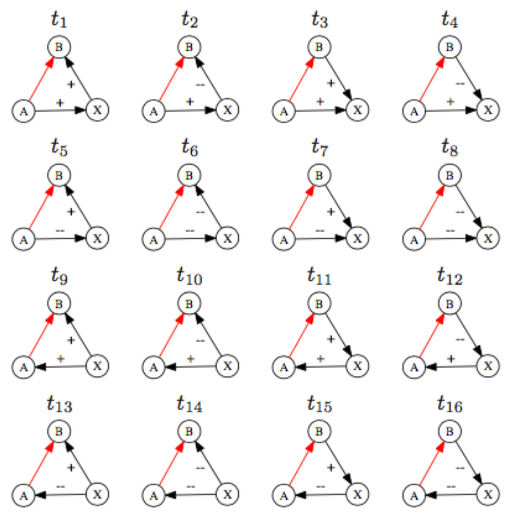

Figure 5: Triad possibilities for two signed edges. [LHK14]

Status Triad Statistics

	btc-alpha		btc-otc	
t_i	count	$p(t_i)$	count	$p(t_i)$
t_1	44946	0.68	61185	0.61
t_2	3763	0.06	6018	0.06
t_3	4320	0.07	5998	0.06
t_4	771	0.01	1592	0.02
t_5	3547	0.05	5307	0.05
t_6	1033	0.02	2124	0.02
t_7	403	0.01	2316	0.02
t_8	871	0.01	1988	0.02
t_9	4276	0.06	5961	0.06
t_{10}	573	0.01	1039	0.01
t_{11}	213	0.00	350	0.00
t_{12}	44	0.00	1476	0.01
t_{13}	1031	0.02	2340	0.02
t_{14}	485	0.01	2554	0.03
t_{15}	170	0.00	162	0.00
t_{16}	13	0.00	69	0.00

The prediction of the red edge in Figure 5 can be made with either balance theory or status theory. With either balance or status theory, surprise is a measure that was developed to determine deviation from the expected labeling of a directed edge.

5 Sign Prediction with Local and Global Methods

For the prediction of edge signs within social networks, Chiang et. al [CHN$^+$14] leverage the local and global aspects of balance theory. They define a measure of social balance based on l-cycles (a path from a node to itself through l edges) in the network and generated a simple prediction rule that we have adapted to weighted, signed networks. A low rank modeling approach was utilized for sign prediction, with the application of matrix factorization for scaling to large networks. With the use of social balance in the global perspective, the accuracy improved significantly for edge sign prediction.

A global approach was developed based on social balance by clustering of complete, weakly balanced networks. Since these networks have low rank adjacency matricies, they can be utilized to solve the sign prediction problem with a reduction to a low rank matrix completion problem. An adjacency matrix of complete k-weakly balanced networks has rank k. This adjacency matrix can be partitioned into $k > 2$ groups such that within-cluster edges are positive and the rest is negative.

Since rank constraint of the low rank matrix completion problem is non-convex, convex relaxation of rank constraint is done with both Matrix Factorization and Singular Value Projection.

5.1 Measures of Social Imbalance (MOI)

Structural balance is constructed as a local approach to sign prediction. A measure of imbalance based on the balance theory, can be formulated in the following manner

$$\mu_{tri}(A) := \sum_{\tilde{\rho} \in SC_l(A)} \mathbf{1}[\tilde{\rho} \text{ is unbalanced}],$$

where $SC_l(A)$ refers to the set of triangles (simple cycles of length-l) in the network A. The edge sign prediction can then be written in the following manner

$$\text{sign}\Big(\mu_l\big(A^{-(i,j)}\big) - \mu_l\big(A^{+(i,j)}\big)\Big) = \text{sign}\Bigg(\sum_{t=3}^{l} \beta_t A_{i,j}^{t-1}\Bigg)$$

with β as the weighting coefficients for unbalanced simple cycles of different lengths.

The Katz measure is then adapted towards Balance Theory for the purposes of edge sign prediction. The Katz measure is an effective proximity score used in link prediction for unsigned networks. The *signed* Katz score has been adopted by Chiang et. al for use in sign prediction. It is defined in the following manner

$$Katz(i,j) := \sum_{l=2}^{\infty} \beta^l A_{ij}^l$$

$$\text{Matrix form: } Katz(i,j) = ((I - \beta A)^{-1} - I - \beta A)_{ij}$$

From the adaptation of the Katz measure, the closed form sign prediction formula can then be written as

$$\text{sign}\Big(\big((I - \beta A)^{-1} - I - \beta A\big)_{i,j}\Big),$$

when $\beta_l = \beta^{l-1}$ with β small enough ($\beta < 1/||A||_2$).

5.2 Higher Order Cycles (HOC)

Another local method we consider in edge sign prediction is the use of Higher Order Cycles. Adapting the approach developed by Leskovec et al [LHK10], there are four possible cases a given edge $e = (i,j)$ can take, incorporating direction and edge sign. When considering the edge between j and k, there is also the same 4 possible configurations. This means there is a total of 16 possibilities for the edge e when considering the number of common neighbors k in each of the 4 $\times$ 4 = 16 configurations.

These configurations are utilized in a supervised variant of the k-cycle method for $k = 3$. We construct A^+ and A^- as the matrices of positive and negative edges such that $A = A^+ + A^-$. These matrices are then trained with logistic regression to predict edge sign.

5.3 Singular Value Projection (SVP)

From the use of MOI, the length of l-cycles can be increased to provide a more global view of the balance structure. From a local point of view, a network is weakly balanced if all triads are weakly balanced. Whereas from a global point of view, a network is weakly balanced if the global structure is composed of all positive edges or the nodes can be divided into k clusters such that all the edges within clusters are positive and all edges between clusters are negative.

Since weakly balanced networks have "low-rank" structure, the edge sign prediction problem can be formulated as a low rank matrix completion problem. This algorithm attempts to solve low-rank matrix completion efficiently. The SVP algorithm is formulated as follows

$$\begin{aligned} \text{minimize } & ||\mathcal{P}(x) - A||_F^2 \\ \text{s.t. } & \text{rank}(X) \leq k, \end{aligned}$$

where the projection operator $\mathcal{P}$ is defined as:

$$(P(X))_{ij} = \begin{cases} X_{ij}, & \text{if } (i,j) \in \Omega \\ 0, & \text{otherwise.} \end{cases}$$

To optimize the minimization problem, the SVP algorithm iteratively calculates the gradient descent update $X^{(t)}$ of the current solution $X^{(t)}$, and projects $\hat{X}^{(t)}$ onto the non-convex set of matrices whose rank are at most k using SVD. After the SVP algorithm terminates and outputs $\bar{X}$, one can take the sign of each entry of $\bar{X}$ to obtain an approximate solution for minimizing the rank of the completed matrix. From the results, we can observe that SVP performs well in recovering weakly balanced networks. However, SVP does not scale very well to large datasets because they require computation of the SVD.

5.4 Matrix Factorization with Alternating Least Squares (ALS)

Another global method we explored is the use of a gradient based matrix factorization approach for edge sign prediction. For a matrix factorization approach, a problem such as the one below is solved with an alternating minimization algorithm that is non-convex.

$$\min_{W,H\in\mathbb{R}^{n\times k}} \sum_{(i,j)\in\Omega} (A_{ij} - (WH^T)_{ij})^2 + \lambda||W||_F^2 + \lambda||H||_F^2$$

ALS solves the squared loss problem by alternately minimizing W and H of a non-convex, minimization algorithm. When either W or H is fixed, the optimization problem becomes a least squares problem with respect to the other variable.

5.5 Signed Prediction Results

Both the bitcoin-alpha and bitcoin-otc graphs have edge weights with a magnitude outside the range of $[-1, 1]$. These edges weights were replaced with edge signs so that all four algorithms can be executed for edge sign prediction.

As can be seen in the table below, the global methods of edge sign prediction (ALS and SVP) performed substantially better than the local methods (MOI and HOC).

		btc-alpha		btc-otc	
		Average	Std Err	Average	Std Err
MOI	Accuracy	0.9158	0.0025	0.9086	0.0007
	False Positive Rate	0.6009	0.0152	0.3981	0.0075
HOC	Accuracy	0.7631	0.0088	0.7711	0.0038
	False Positive Rate	0.2908	0.0125	0.2429	0.0066
SVP	Accuracy	0.9845	0.0119	0.9265	0.0184
	False Positive Rate	0.0694	0.0422	0.2216	0.0543
ALS	Accuracy	**0.9936**	0.0005	**0.9901**	0.0003
	False Positive Rate	0.0236	0.0038	0.0249	0.0013

6 Weight prediction

In this section, we turn our attention of edge weight prediction. Kumar et. al. [KSSF16] devised an edge weight prediction algorithm based on “fairness” and “goodness” scores of each node. We will also pose this problem as a supervised learning task. We adapt the work of Sá et. al. [DSP11], who posed this problem as a supervised learning task with feature vectors based on the structural properties of edges. However, their approach is suitable only for undirected, weighted graph.

In this section, we will extend their approach to weighted, directed networks. Towards this end, we view the edges incident at some vertex in the graph being composed of two different sets of edges: incoming and outgoing edges and accordingly split a vertex v into two vertices v_i and v_o with incoming edges incident at v_i and outgoing edges incident at v_o. Refer to Figure 6.

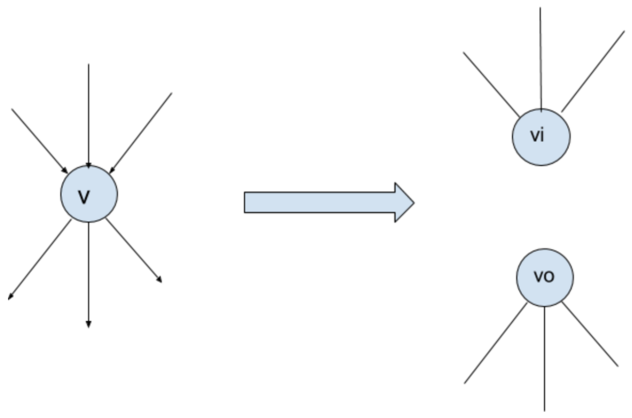

Figure 6: Graph transformation

The above transformation is motivated by the fact that the trust reciprocity rate is very low, for e.g., for bitcoin-otc graph, Figure 7, plots the difference in reciprocal ratings.

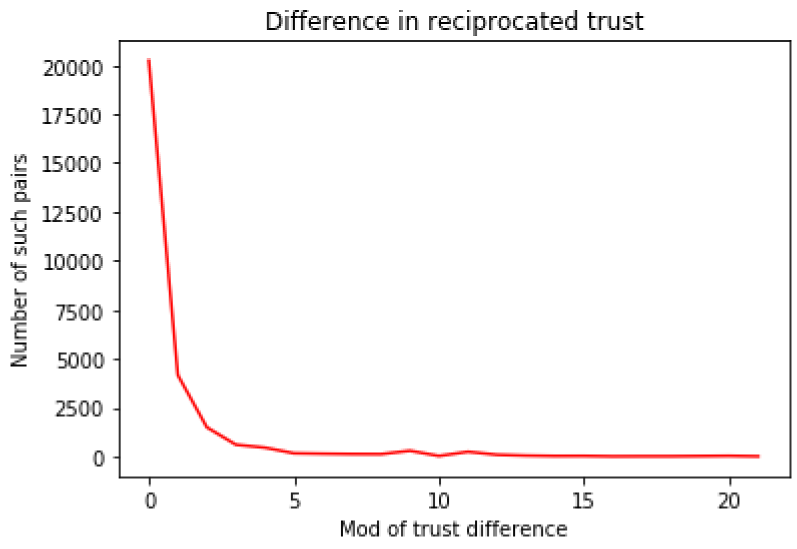

Figure 7: Reciprocation in trust

Let the weighted adjacency matrix of $G = (V, E)$ be denoted by W and outgoing neighbors of v be denoted by $\Pi(V)$ (as defined previously). Extending the work of [DSP11] we define the following metrics for graph which will be used for edge weight prediction:

1. **Common neighbors (CN)** CN measures the strength (number) of common neighbors. This is simply the weighted sum of weights of edges to common neighbors. Even an edge (x, y), the pair of edges to a common neighbor z can be of four types: (outgoing, outgoing), (outgoing, incoming), (incoming, outgoing) and (incoming, incoming). Correspondingly, we

have four features for common neighbors. Mathematically,

$$\begin{aligned}
CN_{o,o}(x,y) &= \sum_{z\in\Pi(x)\cap\Pi(y)} W_{xz} + W_{yz} \\
CN_{o,i}(x,y) &= \sum_{z\in\Pi(x)\cap\Gamma(y)} W_{xz} + W_{yz} \\
CN_{i,o}(x,y) &= \sum_{z\in\Gamma(x)\cap\Pi(y)} W_{xz} + W_{yz} \\
CN_{i,i}(x,y) &= \sum_{z\in\Gamma(x)\cap\Gamma(y)} W_{xz} + W_{yz}
\end{aligned}$$

Here, $\Gamma(x) = \{z : x \in \Pi(z)\}$, *i.e.*, the set of incoming neighbors of x.

2. **Jaccard's Coefficient (JC)** This is similar to CN except that it assigns higher values for edges sharing higher proportion of common neighbors relative to total number of neighbors. We will use the definition of Jaccard Similarity presented in [T+06] for weighted networks and extend it to directed weighted networks:

$$JC(x,y) = \frac{\sum_{z\in\Pi(x)\cap\Pi(y)} W_{xz} + W_{yz}}{\sum_{a\in\Pi(x)} W_{xa} + \sum_{b\in\Pi(y)} W_{yb}}$$

We extend this definition to define four different variants similar to CN.

3. **Preferential Attachment (PA)** PA assumes that probability (weight) of a new link creation is proportional to node degree (weight), *i.e.*,

$$PA(x,y) = \sum_{a\in\Pi(x)} W_{xa} \times \sum_{b\in\Pi(y)} W_{yb}$$

We again define this for all four cases.

4. **Adamic-Adar Coefficient (AA)** Adamic and Adar coefficient is similar to Jaccard's coefficient except that it gives higher importance to common neighbors which have fewer neighbors. Mathematically,

$$AA(x,y) = \sum_{z\in\Pi(x)\cap\Pi(y)} \frac{W_{xz} + W_{yz}}{\log(1 + \sum_{c\in\Pi(z)} W_{zc})}$$

5. **Resource Allocation Index (RA)** This is defined similar to Adamic-Adar Coefficient.

$$RA(x,y) = \sum_{z\in\Pi(x)\cap\Pi(y)} \frac{W_{xz} + W_{yz}}{\sum_{c\in\Pi(z)} W_{zc}}$$

6. **Local Clustering Coefficient (CC)** We use the definition of clustering coefficient for weighted graph presented in [SKO+07]:

$$C(x) = \frac{1}{|\Pi(x)|(|\Pi(x)| - 1)} \sum_{yz} (\hat{W_{xy}} \hat{W_{xz}} \hat{W_{yz}})$$

where $\hat{W}$ are the normalized weights, *i.e.*, $\hat{W} = \frac{W}{max(W)}$. For an edge (x,y), its clustering coefficient is the sum of clustering coefficients of its vertices: $CC((x,y)) = CC(x) + CC(y)$. Note that local clustering coefficient here does not take into account direction of edge.

Finally we train our linear regression model based on the features learnt from the training set edges. Since the graph grows over time, we try to predict the weight of a new incoming edge. The training set is comprised of the remaining edges in the graph.

The results presented here are for bitcoin-otc but a similar analysis can be carried out for bitcoin-alpha as well. Overall, this approach correctly predicted the edge trust weight for 72% **of edges**. Figure 8 describes the results of these experiments. As shown here, the linear regression does well on trust edges whose trust values are the most common.

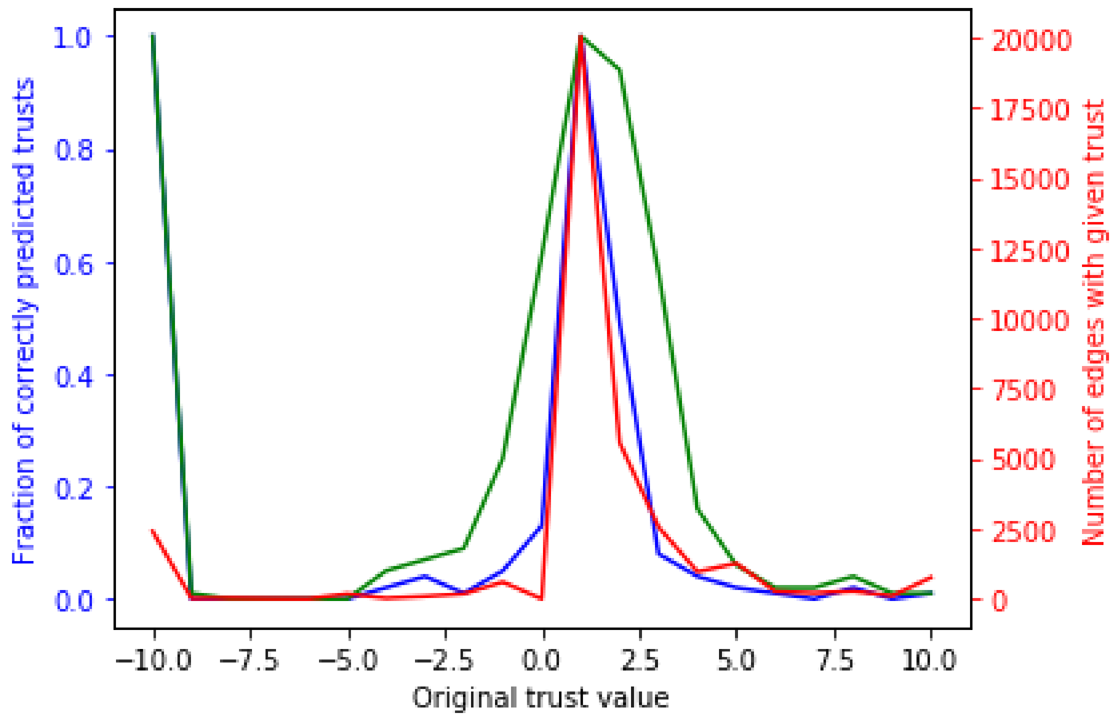

Figure 8: Graph depicting results of linear regression analysis. Here, x-axis is the original trust value. Red line refers to the number of edges with that trust value. Blue line refers to the fractional of edges for which the trust value was correctly predicted. Green line refers to the fraction of edges for which trust value was at most 1 off from the original trust value.

7 Conclusion

Most of the analysis surrounding the trust networks of Bitcoin has been around network analysis of its structural properties and the two theories, balance and status. From the foundation of these two theories, we explored recent work dealing with edge sign prediction from a local and global perspective. The local methods being Social Imbalance and Higher Order Cycles. The global methods being Singular Value Projection and Matrix Factorization with ALS. Our experiments demonstrated both global methods did much better than the two local methods on the two bitcoin graphs.

Then, we extended this work to explore the problem of sign/weight prediction. Working from current state of the art, we defined metrics (such as Clustering Coefficient, Adamic-Adar Coefficient, etc.) for weighted signed networks. From these metrics, we computed the clustering coefficients, predicted edge sign, and also predicted edge weights. The proposed algorithm for edge weight prediction is built on linear regression. This algorithm correctly computes weights (trust) of 72% of edges in the testing set.

References

[CHN$^+$14] Kai-Yang Chiang, Cho-Jui Hsieh, Nagarajan Natarajan, Inderjit S Dhillon, and Ambuj Tewari. Prediction and clustering in signed networks: a local to global perspective. *The Journal of Machine Learning Research*, 15(1):1177–1213, 2014.

[DSP11] Hially Rodrigues De Sá and Ricardo BC Prudêncio. Supervised link prediction in weighted networks. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2281–2288. IEEE, 2011.

[DW13] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *13-th IEEE International Conference on Peer-to-Peer Computing*, 2013.

[FKP14] Michael Fleder, Michael Kester, and Sudeep Pillai. Bitcoin transaction graph analysis. *MIT*, 2014.

[IE07] Antoniou Ioannis and Tsompa Eleni. Statistical analysis of weighted networks. *arXiv preprint arXiv:0704.0686*, 2007.

[KLB09] Jérôme Kunegis, Andreas Lommatzsch, and Christian Bauckhage. The slashdot zoo: mining a social network with negative edges. In *Proceedings of the 18th international conference on World wide web*, pages 741–750. ACM, 2009.

[KSSF16] 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.

[LHK10] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*, pages 641–650. ACM, 2010.

[LHK14] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Signed networks in social media. *ACM*, 2014.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[OKH13] Micha Ober, Stefan Katzenbeisser, and Kay Hamacher. Structure and anonymity of the bitcoin transaction graph. *Future Internet*, 2013.

[RS12] Dorit Ron and Adi Shamir. *Quantitative Analysis of the Full Bitcoin Transaction Graph*. The Weizmann Institute of Science, 2012.

[SKO$^+$07] Jari Saramäki, Mikko Kivelä, Jukka-Pekka Onnela, Kimmo Kaski, and Janos Kertesz. Generalizations of the clustering coefficient to weighted complex networks. *Physical Review E*, 75(2):027105, 2007.

[T$^+$06] Pang-Ning Tan et al. *Introduction to data mining*. Pearson Education India, 2006.

[TCAL16] Jiliang Tang, Yi Chang, Charu Aggarwal, and Huan Liu. A survey of signed network mining in social media. *ACM Computing Surveys (CSUR)*, 49(3):42, 2016.