External influence on Bitcoin trust network structure

Edison Alejandro García, *garcial@stanford.edu*
SUNetID: *06116715*
CS224W - Analysis of Networks

***Abstract*—Networks can express how much people trust or distrust each other. Trust between people may be influenced by factors such as relationship, reputation status, experience, stereotypes, media, information and governments between some. Bitcoin trust networks express how much people trust or distrust each other. Trust in Bitcoin trust network may be influenced by external factors like Bitcoin prices or breach of Bitcoin exchanges like Mt. Gox. One of the goals for this project is to find the effect of external influence to changes in the trust network. One hypothesis is that increase in positive external influence would increase the overall trust in the network, while a drop would decrease the trust. The change in prices may also change the network structure, for example, an hypothesis is that a drop in price would make the positive trust clusters turtle-up [1] to support each other. The second goal would be to create a model that can correctly predict the effects of a network property change in Bitcoin's price.**

I. Introduction

A signed social network (*SSN*) is a network where edges may be labeled as being "positive" or "negative". For instance, if a vertex u dislikes a vertex v, there may be an edge with a "negative" edge label whereas if u likes v, the same edge labeled would be "positive". However, in the real-world, people may like or dislike one another with varying levels of intensity. Person A might dislike B a little bit, but dislike C a lot more. Or person A may trust B a little bit, but trust C a lot more. Or person A may disagree with B a little bit, but disagree with C a lot more. All of these concepts (liking, trusting, agreeing) are different and not necessarily symmetric, yet they all can be captured via (directed) weighted signed networks (*WSN*s).

The Bitcoin OTC trust network (*BTC_OTC*) is a *WSN*. The meaning of positive and negative edges varies from network to network [2]. There may be like/dislike, trust/distrust or agree/disagree relations between people which we don't know about or are yet to form. In this project we treat edge signs as trust indicators, positive edges indicate trust and negative distrust. There is some work on predicting the weight of edges in real-world *WSN* datasets [**?**] and how external events are associated with a network's change in structure and communications [2]. In our study, we start by anaylising how external events influenced the network structure. We have chosen external shocks events [3], which are events that are extreme relative to average events, or unexpected. We don't focus on a set of events but a single one: price shock events. Furthermore, we are aware of other external events that can influence the network structure, like for example news.

One factor that limit the effectiveness of deriving precise network structure is the incomplete node network knowledge. By incomplete we meant that we don't have the all the knowledge of when a node arrived or left the network. What we know is when an edge between two nodes changed it weight. Therefore we don't know, for example, when a node has left the network after seeing an edge.

As part of this project, we will study the network properties overtime under the influence of one external events, price changes, on the *BTC_OTC* network. Furthermore, to create a model that can correctly predict the effect (e.g., change in clustering coefficient, trust, number of nodes, edges) of a "future" change in the Bitcoin network.

II. Related Work

A. Edge Weight Prediction in Weighted Signed Networks

The study proposes two novel measures of node behavior: 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 likability or trust level [2]. Hence, the ratings given by unfair raters should be given low importance, while ratings given by fair raters should be considered important. On the other hand, higher goodness implies the vertex is more trustworthy in the network. Therefore, a 'good' or 'trustworthy' vertex would receive many high positive ratings from fair vertices, while a 'non-trustworthy' vertex would receive high negative ratings from fair vertices. We now provide two equations to compute the fairness and goodness metrics in a mutually recursive manner. For goodness:

$$g(v) = \frac{1}{\mid in(v) \mid} \sum_{u \in in(v)} f(u) \times W(u,v) \tag{1}$$

and for fairness:

$$f(u) = 1 - \frac{1}{\mid out(u) \mid} \sum_{v \in out(u)} \frac{\mid W(u,v) - g(v) \mid}{R} \tag{2}$$

where $\exists u, v \in V$ and $W(u,v)$ is the weight for the edge $(u,v) \in E$. Fairness scores always lie in the [0, 1] interval and goodness scores lie in the [-1,1] interval which is the domain of the edge weights in our case. The maximum

possible difference between an edge weight and goodness score is the range of difference, R = 2[1]. We implemented (1) and (2) as part of our basic network structure analysis tool. Moreover, we studied how fairness and goodness changes over time when price shocks exists.

B. Social Networks Under Stress

They presented a study on how external events are associated with a network's change in structure and communications. To define the extremeness and unexpectedness of price shocks, they defined for each stock s and day d, $a_{s,d}$ and $b_{s,d}$ to be the opening and closing prices respectively of stock's on day d. Furthermore, they defined a delta function as:

$$\Delta_{s,d} = \frac{b_{s,d} - a_{s,d}}{a_{s,d}} \tag{3}$$

which is the proportional change in the price of s on day d.

Two main questions were presented:

1) Explores links between external factors and network structure
2) Asks whether these changes to the network structure can yield additional insight into the behavior of the organization

One of the presented findings is when price shocks occur the communication network tends not to display structural changes associated with adaptiveness. Rather, the network "turtles up". Turtles up means that the network displays a propensity for higher clustering, strong tie interaction, and an intensification of insider vs. outsider communication. In their analysis of network, recovery indicates that most network properties return to their average value one or two days after the shock, suggesting that normalization in relation to these shocks is relatively fast acting. Time was defined as a parameter to evaluate the network changes. We investigate how continuous values of $\Delta_{s,d}$ and discrete price shocks relate to the properties of a graph.

III. Datasets and Features

For the project we will use the Bitcoin OTC trust network[2] (*BTC_OTC*). The latter is composed as:

Dataset statistics	Value
Nodes	5,881
Edges	35,592
Range of edge weight	-10 to 10
Percentage of positive edges	89%
First edge date	2010-11-08
Last edge date	2016-01-25

Figure1 presents the network structure at the date of the last created edge.

Furthermore, we have data mined Bitcoin prices that start on the day of the first edge in the graph and finish on the last day of the graph. As previously mentioned, one of the challenges with the *BTC_OTC* is the notion of which nodes are or not at a point in time in the graph. New edges means that a change in the edge weight happened but it does not says whether we have an edge already. The same applies for the nodes, when an edge is created we don't know if a node just enter the graph or was there already. Based on the latter we decided that a node enters the graph when we see it first edge created. Moreover, we assume that a node and an edge will stay for the entire time in the graph. However, we are still unable to compute if a graph is smaller at any point of time because we assume that nodes will stay in the graph. One possible solution could be to assume a node left the graph when was the last time seen but this still presents other challenges like: what happens when a node was just seen once. The same concepts are applied to any subset of the graph.

We've chosen snap [4] and networkx [5] to work with the data. Snap provides a good set of of algorithm to work with directed graphs, and networkx with good type of graph attributes abstractions. We have found that working with both is an efficient way to derive and maintain, in a short time, network properties.

[1]If edge weights and goodness range over [-1, 1], then $R = 2\ell$.

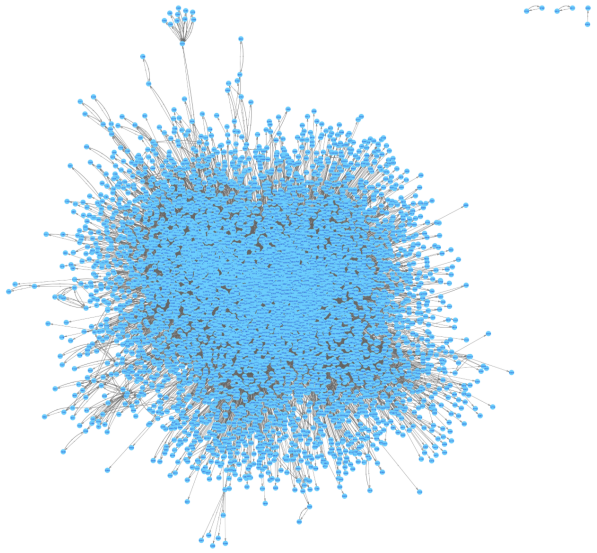

Figure 1. Bitcoin OTC Network in 2016-01-25

IV. Measures

Our primary interest are the subgraphs of the larger network designated by mentions of particular edges at particular points in time. Therefore we have defined a set of network properties of interest based on the related work findings. Furthermore, we include *Fairness* and *Goodness* in our measure set. The goal is to obtain different data signals to help us predict other network properties.

We present, in the the next sub sections, all measure definition with their respective use case in this project.

A. Network Properties

There are several network properties that have been part of the related studies, and from them we studied the following set:

- Number of nodes: $|V|$
- Number of edges: $|E|$
- Ratio of positive weight edges: $(E^+/|V|)$. Weights were defined as trust given from a node to another node. Therefore the ratio of positive edges tell us how trustful is the network.
- Number of negative weight edges: $(|E^-|)$.
- Average Cluster Coefficient: (*C*). It help us identify network structures like turtle-up.
- Number of triads, closed related to C.
- Graph diameter: (*h*). It tell us the longest shortest path in the graph or per SCC
- Number of Strongest Connected Components (*SCC*). We are interested to zoom-in and identify possible network property or structure changes in all SCCs.
- Number of nodes in the largest SCC: $|V_{scc}|$.
- Average Cluster Coefficient in the largest SCC: $|C_{scc}|$.

B. Fairness and Goodness

In section III we have defined the equations for *Fairness* and *Goodness*. We can find the algorithm implementation in Appendix A. In the *Goodness* formula(1) for a vertex v, the incoming edge weights are weighted by the *Fairness* of the vertices that are rating it, so that ratings by fair vertices are considered important. The average of these products over all predecessors yields the goodness of v. When calculating *Fairness* of a vertex u, the smaller the difference between the actual edge weight and the *Goodness* of the recipient, the more fair the vertex. Again, an average for all the ratings given by vertex u is used to calculate the *Fairness* of u[2]. Next we present a high level pseudo-code of the fairness and goodness algorithm:

- Initialization, For all nodes set fairness to 1 and goodness to the in_degree weight divided the number of in_degree nodes
- set iteration number to zero
- goodness score for each vertex is updated using the fairness scores from the previous iteration
- fairness scores are updated using the newly updated goodness scores in the same iteration
- fairness and goodness scores are mutually recursive and are updated till both the scores converge
- algorithm converges when the change between fairness and goodness scores in consecutive iterations for all vertices is less than an error bound ϵ, which we set to 0.001
- scores of fairness and goodness from the last iteration are the final scores

C. Price Shock

We defined two price shock measures:

1) *day price shock* with the formula defined in Eq. (3). We have a *price shock* if the price change (positive or negative) is greater or equal to: 0.05, 0.10, 0.15, 0.20 and 0.30. We study the latter set of percentage changes to better understand if the network properties are also influenced with by how much the price has changed. *BTC_OTC* is a user's trust network and therefore one of the potential external factors to study over the network is the price change. This is our main external influence factor study in *BTC_OTC*.
2) *time range price shock* which is the longest number of days between day zero (d_0) and day one (d_1) with a price change ratio greater than 0.5. To better visualize it, let's say we plot all Bitocoin prices over time, with time on the x-axis and price on the y-axis. Let's say trace a line between d_0 price (p_0) and d_1 price (p_1), by calculating the slope we return the price change ratio $= \frac{(p_1-p_0)}{(d_1-d_0)}$.

V. Methods

BTC_OTC is a trust network where nodes create connections to another nodes and assign a trust rating value to the connection. The trust rating can be updated during the network life spam. However, we don't have any information about what caused such ratings and how they affected the network structure. All related work, mentioned in this paper, have studied how ratings affected the graph structure but our primary goal is to identify if *price shocks* are a possible cause of changes in the network structure.

We are interested in generating subgraphs of the larger network designated by mentions of particular *price shock* at particular points in time. Therefore for a particular *price shock* day we define a directed $G' \subset G$ where G is the network structure on the last day of the graph as defined in section III.

We start building on top of the related work and therefore we first focus on matching their results. First, we run over all time data collecting the network properties information as described in section IV-A. Moreover, we included *Fairness* and textitGoodness which are defined in section IV-B and study turtle-up [1] based on *Avgerage Cluster Coefficient* defined in section IV-A. The strengths and weaknesses of individual algorithms are discussed and their contributions are experimentally evaluated in section VI-B below. We further identify the most relevant structure changes with a series of plots and tables. Next we introduce the baselines and three main approaches.

A. Daily Price Shock

The first part is to pin-point all *price shocks* from the Bitcoin prices data. The *price shock* percentage changes are define in section IV-C. Then we generate subgraphs until the day of the price shock and obtain all network properties.
The idea is to measure network property changes on the exact day of the *price shock* with respect to the end state of the network properties. What we ask is: which network properties changed that day? and how are those properties changes reflected in the final state of the network? Are they empirically significant?
Despite of the good idea this approach will not lead us in the right direction because it only let you compare it with the graph last day structure. A *price shock* can be, as we will see in VI-B section, very early in the graph time and early can be a bit unstable and therefore premature to derive or have a clear view for a statement. Nevertheless, it provided valuable insights on which *price shock* percentages we see the structure most influential changes.

B. Days in Range of Price Shock

We follow section V-C approach but also taking in consideration three days before to the *price shock* day and three days after. Therefore we study seven days of network property changes based on what happens before and after the *price shock* day. This metric gives a better view of what were the network properties before the *price shock* and if they changed during and after the price shock. The latter is one of the study approaches presented in [1]. This is our main external influence factor study algorithm for *BTC_OTC*.

C. Time Range of Price Shock

Now we take a different approach and study the network over a time range with continuous *price shocks*. However, we use the three days before to the price shock day and three days after approach as in *days in range of price shock* method.
Moreover, we choose those time ranges with positive *price shocks* changes follow immediately by a time range with negative *price shocks*. The reason for the positive-negative approach is to study if the negative time range *price shock* presents different network structure than for the positive time range. If we would have set those not consecutive intervals then the analysis will be the same as for *days in range of price shock*.

D. Goals

There are several criteria on how to measure the quality of network structure changes based on external influences like *price shock*. We will focus on the following:

1) Measure network last day structure properties.
2) Calculate network structure properties variance on price shock day.
3) Measure network property variance over price shock percentage.
4) Measure network property variance from three days before and three days after a price shock.
5) Measure network property variance from three days before and three days after over price shock percentage.
6) Define other possible influences for network structure property changes

Moreover, the experimental goal would be to predict a network property change based on the external influence of *price shock*. For the latter we use all previously defined points and make evaluations over time. However, our focus is not to effectively predict a property change but to also create a series of network structure analysis to understand if *price shocks* are a true external network structure influence.

VI. Experimental Setup/Results/Discussion

A. Experimental Setup

As first part, our constant experiments parameters are: Intel i7 4 cores and a single processor at 2.5 GHz with 16GB RAM, Mac OS and Matlab *R2016b*.

1) Primary Factors: These are summarized in Table I and are the only parameters for our study. Nevertheless the full list of *Primary Factors* and their possible *levels* can be found in the source code.

Primary Factor	Levels
Time	['2010-01-01' ... '2016-01-25']
Price Shock	[0.05, 0.1, 0.15, 0.2, 0.25, 0.3]

Table I
Experimental parameter values

2) Secondary Factors: Those consists of hardware configuration. We have tested our optimal solution on better computers and we have seen an improvement on the final *running time*. The hardware configuration was definitely not an influencing factor for our study.

B. Results

This section presents a series of experiments starting with the initial structure network analysis from the last created edge in the graph *2016-01-25*. Then we follow with a *day price shock* and *days in range of price shock* network property analysis, and finally a short analysis with *time range price shock* study.

Table II presents our initial network properties values. At this point the most interesting property is the degree distribution, the size of largest Strongly Connected Component (SCC) and diameter. The number of nodes in the SCC account for 80% of the total number of nodes which can be seen in figure 1. Figure 2 presents a Power Law shape degree distribution with a large number of nodes with high degree (> 40). From the size of the SCC and the degree

distribution we can start saying that the connectivity of the graph is high, also supported by the small graph diameter size. The latter also implies that so far the network does not model a Random Graph Model ($G_{n,p}$).
In *BTC_OTC* the Power Law degree distribution has an α value of 1.7493. There are several sources of Power Law graph models but from [6] we know that with $\alpha < 2$ we can have infinite expectations. We will see that the graph is resilient if we take a *price shock* as a random attack. We will not continue discussing which Power Law graph model *BTC_OTC* follows because is not part of our goal of the study.

Network Property	Value
Start Date	2010-11-08
End Date	2016-01-25
Number of nodes	5881
Number of edges	35592
Ratio of positive edges	89%
Number of negative edges	3563
Avg. Cluster Coefficient	0.177
Max. Avg. Cluster Coefficient	0.18
Diameter	9
Number of triads	33493
Max. Diameter	14
Number of SCC	1144
Number of nodes in largest SCC	4709

Table II
NETWORK PROPERTY VALUES

Figures 3 and 4 present the overall graph *Fairness* and *Goodness* network properties. At this point we are able to match [2] findings.

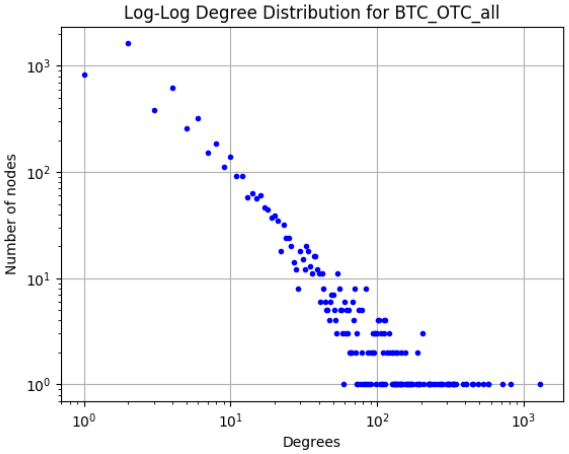

Figure 2. Graph average degree distribution

The total number of shock days, as defined in section V-C is 409 and 60% of those are positive *price shocks*. We continue by introducing a set of results on *price shock* percentage changes. The following plots have a interval

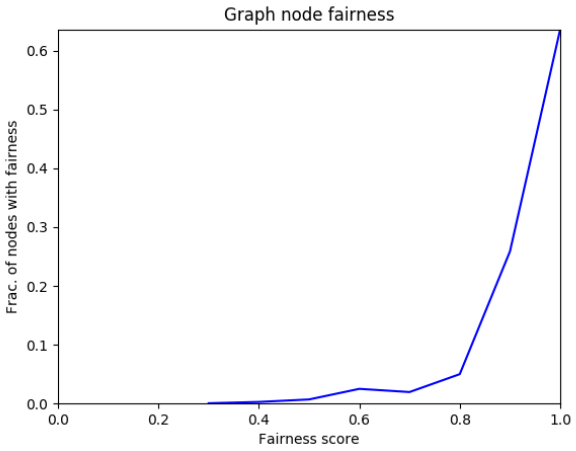

Figure 3. Graph fairness

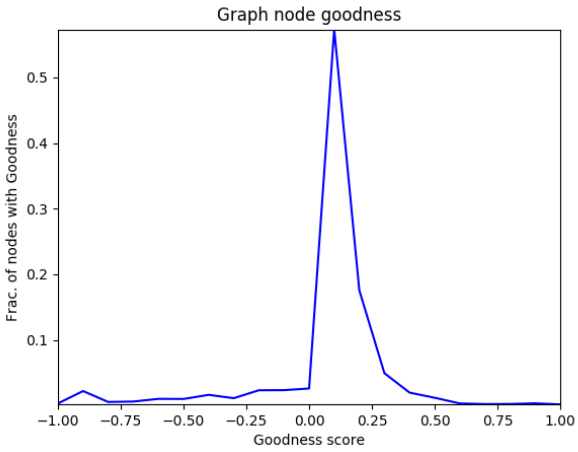

Figure 4. Graph goodness

normalization based on the minimum and maximum size of the network property under study. Normalization is required because the daily results spawn across the whole lifetime of the network and the network continuously evolves over time. For any network property in *BTC_OTC* the value has changed from time t_0 to t_1 where $t_0 < t_1$.

In figure 5 we present our initial results over *price shock* percentages. On the x-axis we have the *price shock* percentage variation. For example: on 0.05 we have accumulated and normalized all dates where the *price shock* is below 0.1 and higher or equal to 0.05 of the price value. Moreover, the analysis returned no differences on the dynamics of the network when we had a positive or negative *price shock*. Therefore, the x-axis only presents a positive *price shock percentage change*.On the y-axis, the normalized variation of the network property. The very first notable information

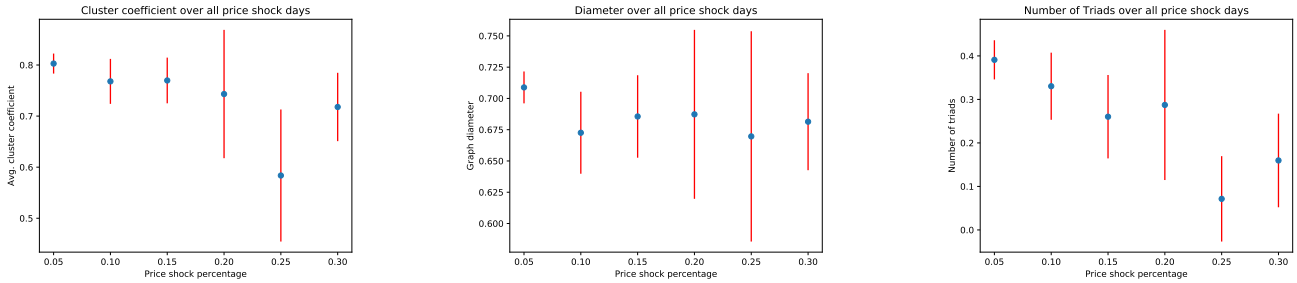

Figure 5. Presents from left to right: cluster coefficient, graph diameter and number closed triads. The red line represent the 95% confidence interval and the blue dot the mean from all aggregated values. On the x-axis we have the price shock percentage change and on the y-axis the network property change.

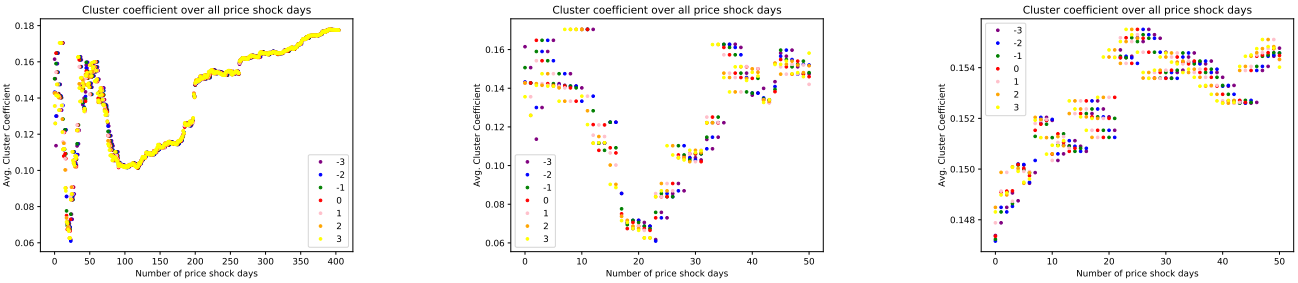

Figure 6. Presents from left to right: cluster coefficient for all price shock dates (enumerate), cluster coefficient for the first 50 price shock dates (enumerate) and cluster coefficient for 200 to 250 dates (enumerate). The values is the legend represent three (-3), two (-2), one (-1) days before the price shock day (0) and one (1), two (2) and three (3) days after the price shock day (0). On the x-axis we have the enumeration of price shock days and on the y-axis the network property change.

are the high variations in some of the confidence intervals. The reason is simply how the dates are distributed across the life spam of the network. In all three plots we can see that for the *price shock* variation percentage of 0.05 we have high confidence deviations. However, for that *price shock* percentage variation the data starts at the beginning of time and end five days before the end of time for the graph. Cluster coefficient is very volatile during the early stages of the graph, figure 6 supports the latest statements.

In figure 6 we have three plots of cluster coefficient, on the x-axis we have the number of dates covered. As we previously said we have detected 409 *price shock* dates with 0.05 or higher price change. On the y-axis we have the property change not normalized. the reason for not normalizing it is that the plot presents the property change over single days. Moreover, it also presents the data for three (-3), two (-2) and one (-1) days before the *price shock* day (0) and one (1), two (2) and three (3) days after the *price shock* (0). In the left most plot we can see that in the early days (0-75 days) of the graph clustering coefficient had high variation across all seven days. while from day 120 on the cluster coefficient normalized and keep on ascending until the end of time. To go a bit more in depth we can see in figure 6 the middle plot for early days (0-50) of the graph and how between the seven days there are differences with respect to the *price shock* day (0). While we move forward in time, the right most graph in figure 6, we see that the values across the seven days start getting tighter without much variation. Figure 5 shows us the network property variations over *price shock* percentages while figure 6 help answer the high confidence variations from figure 5. For space restrictions we don't present the same analysis for diameter and triads. However they follow the same analysis approach and it conclusions. For the network property largest SCC average cluster coefficient $|C_s cc|$ follows the same case given that most of the cluster coefficient changes in figure 5 happened in the largest SCC.

On the related work [1] turtle-up is defined as a network property change based on *price shock*. Our data does not shows any signs of turtle-up nor we can denied it existence. Two of the possible reasons are the different network assumptions and analysis between the two studies. What we can say is: cluster coefficient is not affected by *price shock*. The latter is also supported, as previously mentioned, in [6] as Power Law networks being resilient to random attacks.

For *Fairness* and *Goodness* we have some relevant changes. Despite of the fact that *Fairness* and *Goodness* are not network property structures by themselves they show that something is actually happening on the network at the trust level. We present these findings in order to support the

relevant work and to raise the fact that price changes do actually affect *BTC_OTC* at some network structure level. The important points from figure 7 without overlapping with the related work findings are: for both values there are significant changes over all price shock percentage variations. Moreover, on the second and fourth plots starting from the left there are signs on when those changes happen. The days after the *price shock* are the ones with highest variance, in particular the second and third days. However, the data shows a normalization in the values for *Fairness* and *Goodness* right after the third day.

To continue with the network structure analysis we have figures 8 and 9 with number of nodes $|V|$, number of edges $|E|$ and number of nodes in the largest SCC $|V_scc|$ analysis. The results are the same as previously described for cluster coefficient, figure 8 shows us variations while figure 9 shows no significant variations over the seven days. So far, we have covered a set of network properties without any significant network change that can be link to *price shocks*. For the rest of the properties that we have not yet mentioned, *number of negative weight edges* ($|E^-|$) the collected data returned no findings of high variations that could be influenced by *price shock*. Thus, until this point in the study there are no network properties in *BTC_OTC* that presented significant changes over any *price shocks* percentage changes and over a ranges period of days from the *price shock* day.

Moving forward to *time range price shock*, figure 10 presents the results with respect to *cluster coefficient*, *number of edges* and *Fairness*. Before we continue with the results, the plots x-axis is split in positive and negative *time range price shock* in the following way: we start at 0 with a positive time range follow by a negative time range in x-axis number 1, and so on changing from positive to negative until number x-axis value 4. The dates for the positive time ranges are ['2013-11-29', '2014-01-05', '2014-06-02'] and for the negative time range are ['2013-12-18', '2014-04-10']. Therefore, from '2013-11-29' to '2013-12-18' we have a negative time range. Meaning that the ratio of negative *price shocks* is continuos in that period of time. In figure 10 we have three properties but the analysis returned the same results as for the previously discussed *daily price shock* and *days in range price shock*. We have not found any evidence of network property changes over *price shock* with the current method.

VII. Conclusions

We have developed a novel approach to study *BTC_OTC* network and property changes based on *price shock* by using the related work algorithms as *Fairness* and *Goodness* and extending it with other traditional network analysis algorithms. Our contribution includes developing three methods to compute network properties as defined in section V and an extensive number of experiments to collect all data that lead us to the discussion in the results section. Furthermore, the results demonstrate that *price shock* is not an external influence that changed the network properties under study at any *price shock* date. However, neither we can conclude that other methods will present the same conclusion. In our study, we did not take into account possible network assumptions like in [2]. Therefore, the network property changes prediction over *price shock* goal ends inconclusive and still and open question. Finally, our method allows for a flexible and fast graph property study to better match the demand of various practical applications based on external influence over network properties.

VIII. Acknowledgments

The authors thank Srijan for his time and project ideas and friends for their valuable project feedback.

References

[1] D. M. Romero, B. Uzzi, and J. M. Kleinberg, "Social networks under stress," *CoRR*, vol. abs/1602.00572, 2016. [Online]. Available: http://arxiv.org/abs/1602.00572

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

[3] C. G. Gilbert, "Unbundling the structure of inertia: Resource versus routine rigidity," *The Academy of Management Journal*, vol. 48, no. 5, pp. 741–763, 2005. [Online]. Available: http://www.jstor.org/stable/20159695

[4] J. Leskovec and R. Sosič, "Snap: A general-purpose network analysis and graph-mining library," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 1, p. 1, 2016.

[5] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, Pasadena, CA USA, Aug. 2008, pp. 11–15.

[6] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, pp. 251–262, Aug. 1999. [Online]. Available: http://doi.acm.org/10.1145/316194.316229

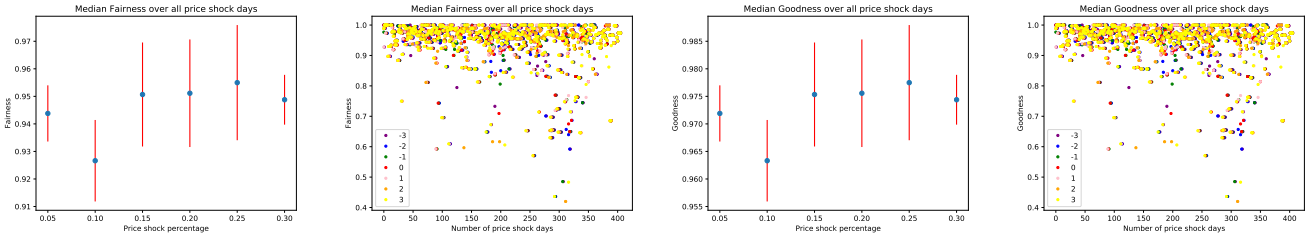

Figure 7. Presents from left to right: Fairness over price shock percentage variations, Fairness per price shock day over seven days, Goodness over price shock percentage variations, Goodness per price shock day over seven days.

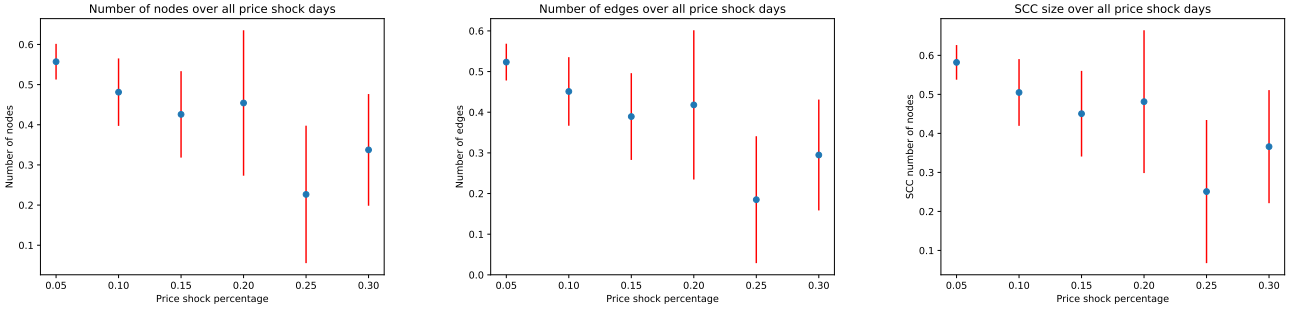

Figure 8. Presents from left to right: number of nodes, number of edges and size of larges SCC in the network. The red line represent the 95% confidence interval and the blue dot the mean from all aggregated values.

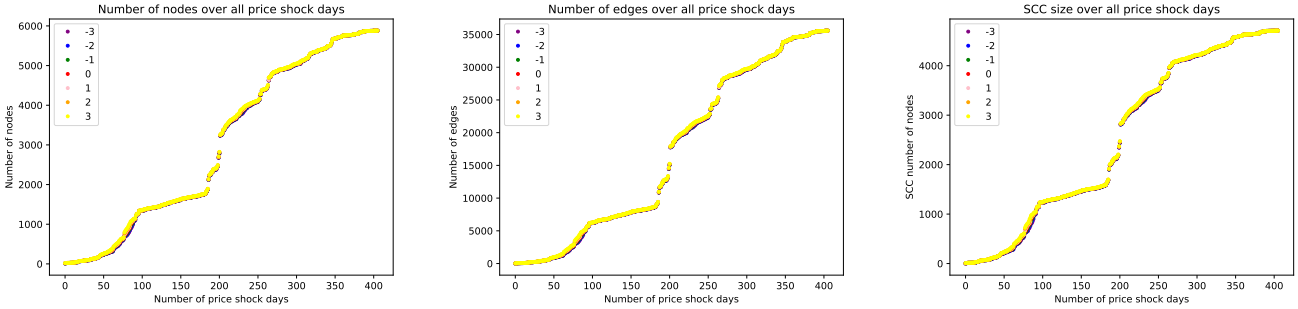

Figure 9. Presents from left to right: number of nodes, number of edges, size of the largest SCC for all price shock dates (enumerate). The values is the legend represent three (-3), two (-2), one (-1) days before the price shock day (0) and one (1), two (2) and three (3) days after the price shock day (0). On the x-axis we have the enumeration of price shock days and on the y-axis the network property change.

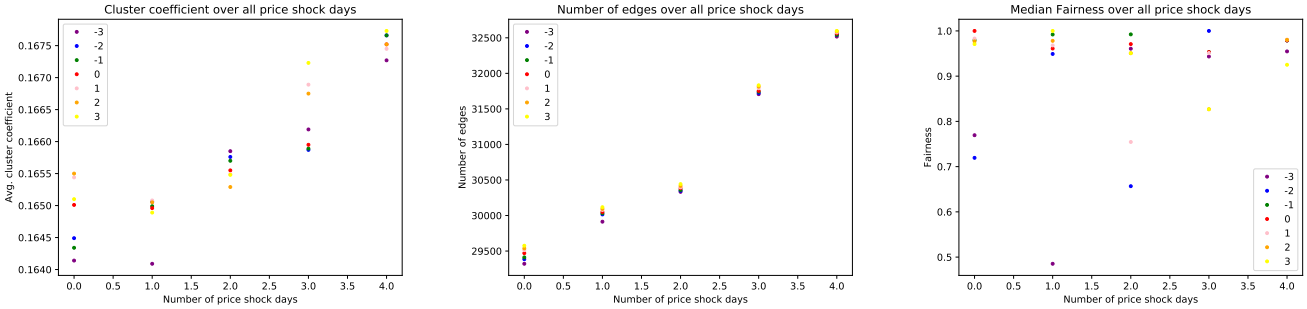

Figure 10. Presents from left to right: number of nodes, number of edges, size of the largest SCC for all price shock dates (enumeration). The values is the legend represent three (-3), two (-2), one (-1) days before the price shock day (0) and one (1), two (2) and three (3) days after the price shock day (0).

APPENDIX

```
def initialize_scores(G):
    fairness = {}
    goodness = {}
    nodes = G.nodes()
    for node in nodes:
        fairness[node] = 1
        try:
            goodness[node] = G.in_degree(node,
    weight='weight') * 1.0 / G.in_degree(node)
        except:
            goodness[node] = 0
    return fairness, goodness

def compute_fairness_goodness(G):
    fairness, goodness = initialize_scores(G)
    nodes = G.nodes()
    iter = 0
    while iter < 100:
        df = 0
        dg = 0
        for node in nodes:
            inedges = G.in_edges(node, data='
    weight')
            g = 0
            for edge in inedges:
                g += fairness[edge[0]] * edge[2]

            try:
                dg += abs(g / len(inedges) -
    goodness[node])
                goodness[node] = g / len(inedges)
            except:
                pass

        for node in nodes:
            outedges = G.out_edges(node, data='
    weight')
            f = 0
            for edge in outedges:
                f += 1.0 - abs(edge[2] - goodness[
    edge[1]]) / 2.0
            try:
                df += abs(f / len(outedges) -
    fairness[node])
                fairness[node] = f / len(outedges)
            except:
                pass

        if df < math.pow(10, -3) and dg < math.pow
    (10, -3):
            break
        iter += 1
    return fairness, goodness
```