Finding vicious loop in Bitcoin trust networks through motifs

Elena Bettella, Andrea Costalonga and Luca Scaramuzza

1 Introduction

People review's are a key measurement to understand the quality of a product. The networks that we are going to consider are who-trusts-whom networks of people who trade using bitcoin on a platform. In order to prevent transactions with fraudulent and risky users, which are anonymous in these nets, there's the necessity of keeping user records with their correspondent reputation gains/losses.

Our hypothesis is that fraudulent users don't act alone, they group up and create small vicious loops trying to fool other users. Our goal will be investigating the possible relation between certain graph motifs and the behavior of users. Motifs in this type of graph are very important for several tasks in the analysis and prediction of the relationship between users. We are going to focus on possible fraudulent patterns, indeed fraudulent users may give fake ratings for excessive monetary gains.

Our work is divided in two parts: first we want to verify that motifs we are going to consider are significant, secondly we want to verify that there is a correlation between some motifs and the presence of fraudulent users.

All the experiments we performed can be found in the following link [1]

1.1 Datasets

Both datasets are who-trusts-whom networks of people trading Bitcoins on two platforms: Bitcoin OTC[3] and Bitcoin Alpha[2].

Members of these platforms rate other members on a scale of -10 (total distrust) to +10 (total trust) in steps of 1. The networks can be described as Multigraphs with multiple directed and weighted arcs; BTC OTC and BTC Alpha are composed of respectively 5,881 nodes and 35,592 edges, and 3,783 nodes and 24,186 edges. In this paper we are going to report only the analysis on the Bitcoin OTC dataset, however since the two datasets are based on the same topic, there could be applied the same analysis also to Bitcoin Alpha.

1.2 Python library

The main library we used is python-graph-tool, since it has a great multiprocessing implementation and bindings in C++. We also used other standard libraries to manage statistics such as numpy and pandas.

2 Motifs Significance check

2.1 Motifs

First let's go through some basic definitions. A graphlet is a small connected subgraph, and its size is equal to the number of nodes involved in it. A motif is a graphlet which is significantly overrepresented in the graph. The following are all the types of graphlet with k=3, with k as the number of connected nodes involved in the subgraph.

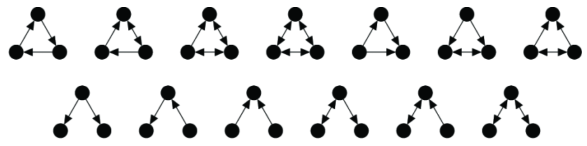


Figure 1: All possible motifs with k=3

We decided not to compute graphlets for k=4 because it was too computationally demanding. Nevertheless our code can be easily modified in order to compute motifs with k greater than 3.
The counting graphlet problem requires finding the number of times a graphlet appears exactly as a subgraph, that is the number of times a graphlet H is isomorphic to the graph G.
Two graphs $G = (V_g, E_g)$ and $H = (V_h, E_h)$ are isomorphic, that is $G \simeq H$, if there exists a bijection function $f\colon V_g \to V_h$ such that $(u, v) \in E_g \iff (f(u), f(v)) \in E_h$.
Since our graph is directed the number of possible graphlet is much larger with respect to an undirected graph, in particular there are 13 possible graphlet with k=3, while for an undirected graph there are just 2.
To compute the number of graphlets we used the function *motifs(graph,k)* available in the graph-tool library. This function is implemented using the ESU algorithm.

2.2 Monte Carlo simulation

In order to assess significance to graphlets we need to compare the results obtained from our graph with other random graphs. A random graph is a graph taken uniformly at random among all graph with $|V|$ number of vertices. To generate them we used the Erdős–Rényi model which preserves the number of edges in expectation, so each edge appears with probability p independently of all other events.
We then implemented a Monte Carlo simulation in order to generate several random graphs.
The simulation worked as follows:

1. 10 random graphs are generated
2. The motif algorithm is performed on each random graph, an array of motifs and their occurrences is returned for each graph
3. A vector m is computed as the mean of the occurrences
4. This array is saved and the routine is repeated several times.

From the Central Limit Theorem we know that a random variable $X = \frac{1}{n}\sum_{i=1}^{n} X_i \sim Z(E[X], std(X))$ given $X_1, ..., X_n$ i.i.d. random variables. In our case we are considering the number of occurrences of a certain motif in different random graphs as i.i.d.'s random variables. The vector m in every iteration can be considered then as a sample from a random vector $\mathbf{Y} \sim [Z(\mu_1, \sigma_1), ..., Z(\mu_n, \sigma_n)]$ with n as the number of possible subgraphs with k=3 in our case.
In our simulation just 7 of these subgraphs appeared with a remarkable frequency ($\mathbf{Y}$ will be a multivariate normal distribution with 7 components).
After this simulation we obtained a collection of samples from which we estimated the distributions for the 7 relevant subgraphs; this statistic will be our null hypothesis H_0.

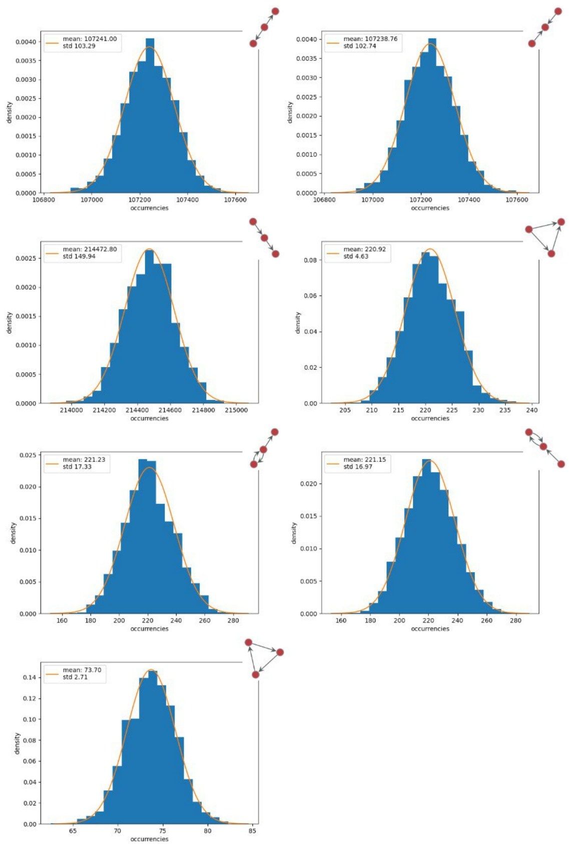


Figure 2: Resulting multivariate distribution obtained from the simulation

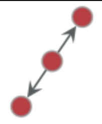
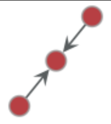
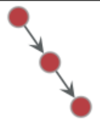
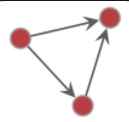
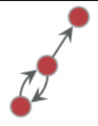
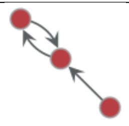
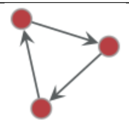
Motif	Mean Variance	z-score p-value
	107236.178 103.095	451.668 ~ 0
	107238.167 104.475	-675.666 ~ 0
	214475.807 151.089	-988.377 ~ 0
	220.958 4.726	737.455 ~ 0
	221.599 17.543	25108.601 ~ 0
	221.550 17.052	9940.567 ~ 0
	73.587 2.684	2.017 0.02

Table 1: Values of mean, variance, z-score and approximated p-value

Our goal is to verify that our graph is very unlikely to come from this distribution. A hint of this assumption can arise from a simple run of the ESU algorithm on our instance that returns 13 motifs instead of 7. Since we are skeptical, we proceeded on calculating z-scores and p-values of our observed sample. The results can be seen in the right table.

As we can see in every case we have for every motif a high z-score (in absolute value) and a low p-value. At this point we are pretty confident in stating that our graph is far from random and that every motif returned by the ESU algorithm is relevant.

3 Second experiment

We will now focus on the identification of fraudulent users. The paper *Edge Weight Prediction in Weighted Signed Networks*[5] describes a nice algorithm to predict the probable behavior of a user in a net based on ratings received and given. The algorithm is based on two heuristics: $fairness(v) \in [0,1]$ is an indicator of how fair the ratings given by the user are, and $goodness(v) \in [-1,1]$ states if a user behaves genuinely or not. The prediction of the weight of a possible edge between two users is done by multiplying the fairness of the source by the goodness of the target.
Here's what comes from a run of the algorithm on the overall graph: we derived a new metric from these two heuristics, which we called trust, that defines the level of trust of a node in the network. The metric is computed starting from fairness and goodness as follows:
trust[node] = goodness[node](1-fairness[node])*
As we can see in figure 4 this function accumulates near 0+ the nodes with high fairness and goodness and in 0- the nodes with negative goodness and high fairness. The nodes that are far from 0 are: in the negative side nodes with negative goodness and low fairness, in the positive side nodes with high goodness and low fairness.
We approached the problem in the following way:

1. We've classified the nodes as fraudulent or not based on the trust metric we've created.
2. We runned the ESU algorithm on the graph, asking to return all the vertex maps (and not just the occurrences of a motif).
3. We counted how many vertex maps were infected by at least one node and two nodes, for every motif.

After the first step of our procedure we gathered some important information that will be presented in the following two plots.

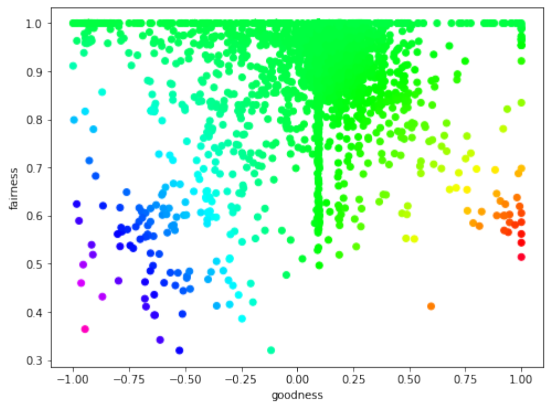


Figure 3: Goodness and fairness embedding

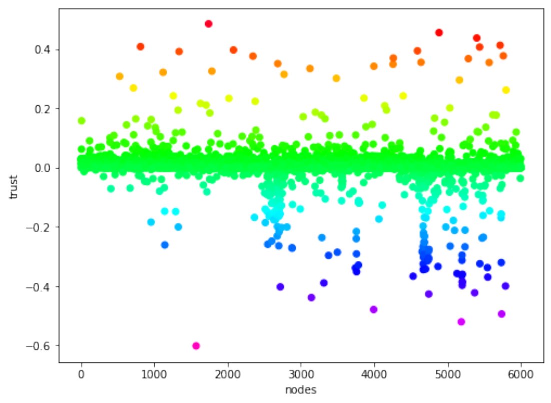


Figure 4: Trust of each node

In the first picture we've used goodness and fairness of every node to create an embedding; as we can see in the bottom left quarter we have some nodes with low fairness and goodness, those are our evil nodes.
In the second picture there is just a plot of the trust measure we've computed for every node.

We set a threshold of -0.1 on the trust measure to mark a node as evil. After executing point 2 and 3 of the procedure described above, we obtained the following picture.

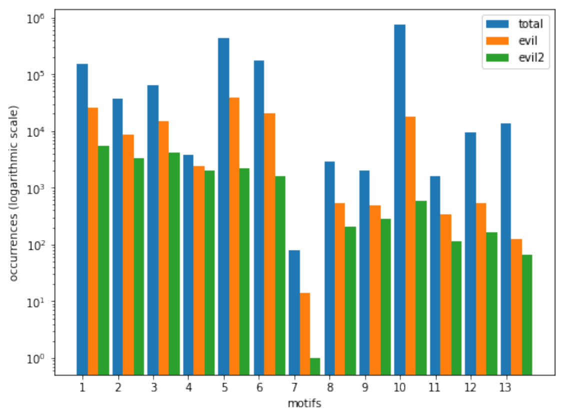


Figure 5: Occurrences comparison

In this figure there is a comparison between the total number of occurrences for each motif, the number of occurrences in which at least one of the vertices involved in the subgraph is evil and the number of occurrences in which at least two of the vertices involved in the subgraph is evil.

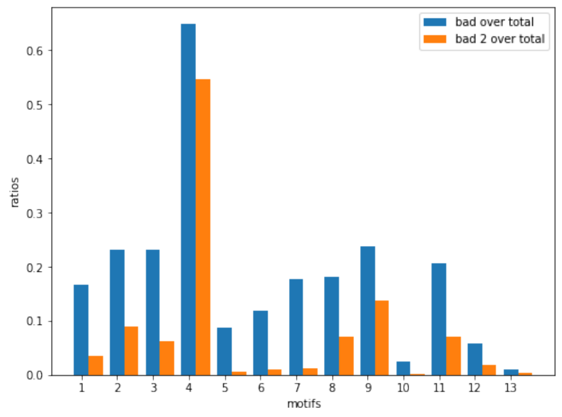


Figure 6: Ratios

We can see it clearer in picture 6 where we compared the ratio between the total number of occurrences and the number of infected subgraphs by at least one and at least two evil vertices, for every motif returned. We can also notice that there is a peak in the ratio for the fourth motif; here we have that more than 60% of the occurrences have at least one evil node, that goes down to 50% when considering at least two evil nodes for occurrence.

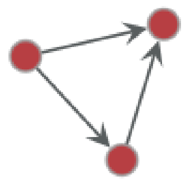


Figure 7: Fourth motif

Still, we don't have any insurance on the fact that the presence of this motif could be a marker on the presence of fraudulent users but, we cannot either negate a possible correlation between them.

4 Conclusion

Summing up, the work we've made consisted in checking the significance of the motifs observed in our network, comparing them with the multivariate distribution of motifs coming from random graphs and in finding whether there exists a strong correlation between fraudulent users and the presence of certain motifs.
Our results were quite satisfactory in the first experiment, which confirmed what we hoped, that the motifs in our network were very different from the ones coming from a random network.
The second experiment wasn't as enthusiastic as

the first one; we can see that there could be a correlation between certain motifs and the presence of fraudulent users (like the motif 4) but we cannot say for sure. More experiments are needed to understand if this phenomenon appears in other similar networks; if it does appear, then we may convene in stating that this correlation exists. By now, what we can say is that deeper work is needed to obtain concrete results but this could be a great starting point for future work, starting from analyzing in the same way also the Bitcoin Alpha dataset.

References

[1] `https://github.com/Starkiller13/LFN_project`.

[2] Bitcoin Alpha trust weighted signed network. Data retrieved from Stanford Network Analysis Project, `https://snap.stanford.edu/data/soc-sign-bitcoin-alpha.html`.

[3] Bitcoin OTC trust weighted signed network. Data retrieved from Stanford Network Analysis Project, `https://snap.stanford.edu/data/soc-sign-bitcoin-otc.html`.

[4] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. Subrahmanian. Rev2: Fraudulent user prediction in rating platforms. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pp. 333-341, ACM*, 2018.

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