***Abstract*—**
***Index Terms*—**

Analyzing the Ethereum Blockchain

Christin Sauerbier
Humboldt University Berlin
Berlin, Germany
christin.sauerbier@outlook.com

Roman Proskalovich
Humboldt University Berlin
Berlin, Germany
romanarion@gmail.com

Thomas Siskos
Humboldt University Berlin
Berlin, Germany
thomas.siskos@hu-berlin.de

I. Introduction

II. Literature Review

III. Ethereum Theory

IV. Ethereum and the Blockchain

V. Methodology

A. Data Preparation

In order to access the data in any form of blockchain technology one must first be an owner of a full node of the respective network. This means that the whole blockchain data with all its records is stored on the researcher's device. However for the most part, data in this form is not stored in a manner that is easy to query. For this reason we opted to become owners of a full node of the Ethereum blockchain, to subsequently scrape the blockchain for the desired information and to store the results in a relational database, which is easy to query.

Algorithm 1 Data Scraper: Overview

```
Check progress file if the ETL can be resumed from a block.
if yes then
    current block c ← content of progress file
else
    c ← 1
end if
s ← 2000000
while c ≤ s do
    Overwrite progress file with c
    content_Block ← getBlock(c)
    row_Block ← extractInfo(content_Block)
    write(row_Blocks, "Block")
    for each transaction in content_Block do
        row_Transaction ← extractInfo(transaction)
        write(row_Transaction, "Transactions")
    end for
end while
```

The data-scraper is written in the `Python` language using the `web3.py` module (Rossum, 1995; Piper & Caver, 2018). Through `web3.py`'s interface it is possible to interact with the Ethereum blockchain. In particular the `web3.py` module enables us to extract information for every block and each transaction that is recorded in the blockchain.

In particular, for each block inside a predetermined range we query its contents and obtain the data that is interesting for the further analysis. On the block level we extract the hash of each block as a unique identifier and the time each block was inserted into the blockchain. Note that this is not necessarily the time that the transaction was being commissioned. Finally we query the total gas that was used for each block as well as the gas limit for each block. Along with that information we retrieve a list of transactions that are contained within that block.

Making use of the list of transactions within a block we extract the wallet addresses of both sender and receiver, as well as the total value, of the transaction. Additionally we retrieve information on the gas price for the transaction. Figure 1 summarizes the relations within that database and Algorithm 1 gives an outline of the datascraper in pseudocode.

B. Graph Analysis

We can characterize a network by a number of metrics. The degree of a node in the network is the number of incoming or outgoing edges. We define a *triplet* i, v, j as an ordered set of three nodes, where v is the focal point and the undirected edges $< i, v >$ and $< v, j >$ form the neighboring edges. A triplet is considered closed, when there are exactly three connections for the three nodes. Three closed triplets form a *triangle* (Meghanathan, 2018). Additionally, we can define multiple measures to investigate a node's role inside the network.

1) Degree Centrality: The most common centrality measure focuses on counting the number of nodes in it's immediate vicinity. It measures the centrality by counting the number of direct connections to the node. For a network with g nodes, the degree centrality of node n is defined as

$$C_D(n) = \frac{k(n)}{g-1}, \tag{1}$$

where $k(n)$ denotes the degree of node n. The Degree Centrality is normalized by $g-1$ since this is the highest possible degree, a node that is connected to every node in the network other than itself (Meghanathan, 2018).

2) Closeness Centrality: While the Degree Centrality only takes the adjacent nodes into account, it may be possible that the most central in this sense, i.e. the node with the highest degree, is in fact not close to other nodes in the network. This shortcoming is overcome by the *Closeness Centrality*. It considers the sum of geodesic distances, i.e. the number of

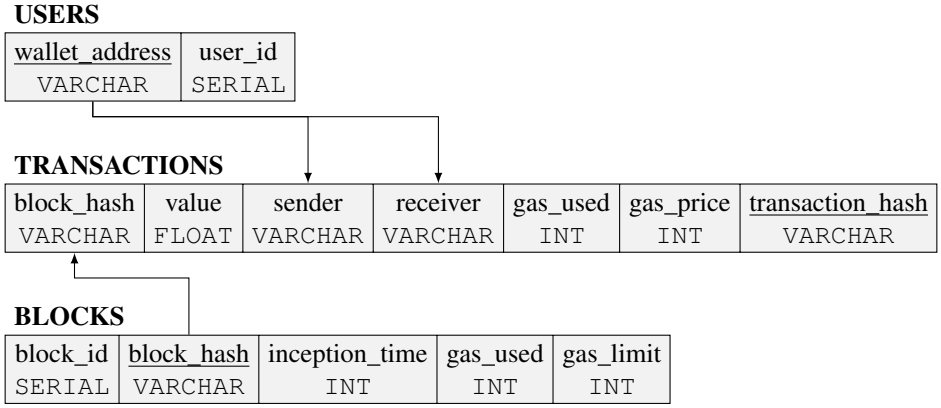


Fig. 1: Database Model: the primary keys of each table are underlined, foreign keys are represented by arrows, the type of each column is capitalized.

edges in the shortest path between two nodes. The Closeness Centrality of a network with g nodes is defined as

$$C_C(n) = \frac{g-1}{\sum_{i=1}^{g} d(n,i)}, \tag{2}$$

where $d(i,j)$ stands for the geodesic distance between node i and j (Meghanathan, 2018).

3) Betweenness Centrality: So far, all the measures focus heavily on the direct influence of the node in question on others. However, two nodes which are not adjacent can also influence each other indirectly through other nodes in the network. The *Betweenness Centrality* highlights the nodes that lie most frequently in the connecting path between two nodes. For a network with g nodes the Betweenness Centrality for node n is defined by the sum over all geodesic distances that pass node n, normalized by the sum of all shortest paths of all pairs of nodes:

$$C_b(n) = \frac{\sum_{j<k} g_{jk}(n)}{g_{jk}}.$$

Here, g_{jk} denotes the number of shortest paths between two nodes of the netwok and $g_{jk}(n)$ marks the number of shortest paths that pass through node n. Since the maximum number of such paths is

$$\frac{(g-1)(g-2)}{2}.$$

One can thus further normalize:

$$C_B(n) = \frac{C_b(n)}{\frac{(g-1)(g-2)}{2}}. \tag{3}$$

4) Clustering Coefficient: In order to account for local structures in graph theory, one uses the *Clustering Coefficient*. There is evidence that suggests that the ties between nodes in most real-world networks tend to create tightly knit groups with a relatively high density of connections (Watts & H. Strogatz, 1998).

There exist two versions of this measure, the *global* and the *local* Clustering Coefficient. The global Clustering Coefficient C_G is the number of triangles over the total number of open or closed triplets in the graph. This measure can be applied to both directed and undirected networks.

The local Clustering Coefficient for a node n is given by the proportion of links between all adjacent vertices divided by the number of links that could possibly occur between them. For a directed graph the in- and outgoing edges have a different meaning and therefore the maximum number of immediately adjacent neighbors of n is $k(k-1)$, where k is the degree of node n. The neighborhood of a node n, i.e. its immediately connected neighbors, is defined as

$$N = \{n : e_{ij} \in E \cap e_{ji} \in E\},$$

where E is the set of edges.

Thus the local clustering coefficient of of node n is given by:

$$\begin{aligned} C_L(n) &= \frac{\text{number of closed triplets centered around n}}{\text{number of triplets centered around n}} \\ &= \frac{|\{e_{jk} : v_j, v_k \in N \cap e_{jk} \in E\}|}{k(k-1)}. \end{aligned} \tag{4}$$

In an undirected graph the meaning of in- and outgoing edges is considered identical. Therefore if a node n has degree k, there could exist a total of $\frac{k(k-1)}{2}$ edges within n's neighborhood. Thus the local Clustering Coefficient of a undirected network can be defined as

$$C_L(n) = \frac{2 \cdot |\{e_{jk} : v_j, v_k \in N \cap e_{jk} \in E\}|}{k(k-1)}. \tag{5}$$

This means that nodes with a high Clustering Coefficient play a large role when relaying information through the network (Knorr, 2019).

5) Power Law and the Small World Phenomenon: The small world phenomenon describes that the size of a network does not have any effect on the size of ties among its nodes.

This means that the path lengths are characteristically small just like random graphs (Watts & H. Strogatz, 1998).

In a random graph the degree of a node n follows a power law distribution, i.e. we can say:

$$P\left(K(n) = k\right) \propto k^{-\alpha}, \tag{6}$$

where $K(n)$ is the random variable that describes the degree of node n and α is a constant whose values range between 1.6 and 3.0 (Newman, 2003). This distribution suggests that most nodes will have a low degree, while a small number of nodes will display a large number of connections (Alstott, Bullmore, & Plenz, 2014).

C. Descriptive Analytics

VI. Findings

A. Data Preparation

We encountered many difficulties during our attempts at collecting and refining the data. At first, we attempted to deploy the data-scraper on a `RaspberryPi 3` Model B in conjecture with an external HDD hard drive. We decided on this option, since it did not require any of us to commit indispensable computing ressources and at that time it seemed to be the most convenient option.

That attempt was, in hindsight, doomed to fail due to the hardware's limitations. The problem did not lie in the hardware intensity of the data-scraper in itself, but rather in the inordinate amounts of memory it takes for `go-ethereum` to synchronize with the Ethereum blockchain. The interface to the ethereum blockchain, `go-ethereum`, is written in a manner that is primarily optimized for mining ether and in order to facilitate the commissioning of transactions and smart-contracts. The way the programm achieves this is by consuming as much memory as there is available on the system, or at least as much memory as the user is willing to allocate to it before starting the program. However, whenever the program runs out of memory or whenever it attempts to exceed its predetermined limit the process shuts down and kills itself. This does not result in a system crash, but it does inhibit the workings of the `Python` libraries that are used inside the data-scraper.

This design of the `go-ethereum` program frequently led to the `RaspberryPi`'s 1 GB of memory being filled after only a short period of time, usually between thirty minutes and three hours. Note, that the data-scraper's construction did anticipate interruptions and it is possible to resume scraping the Ethereum blockchain from the current block without the loss of any data and without the need to redundandly query more than the one block at which the disconnection happened. This behaviour meant that the `RaspberryPi` was idling for most of the time.

Unfortunate as this was, it was not the only problem. The use of an external hard disk drive for storing the database meant that a considerable amount of the time during which the `RaspberryPi` was productive was spent on input/output-operations while it was commiting the scraped information into the database. In order to alleviate the bottleneck that was presented by the I/O-operations it would have been favorable to use an external solid state drive for that task. However, we were too frugal to obtain one just for the singular purpose of this seminar paper.

This means that after running the data-scraper for an entire month, along with taking all the care that is reasonable, in order to keep the scraper afloat, we were able to extract 120.000 transactions from 15.000 blocks which tantamounts to merely three days of data.

We solved all these problems by taking up the offer *Microsoft Azure* provides for students. Using this free initial student credit on the *Microsoft Azure* servers it was possible to set up a virtual machine with 32 GiB of memory along with access to an additional solid state drive. The fact that it is possible to rent out a `Linux` machine on the *Microsoft Azure* servers made it possible to migrate the scripts for the data-scraper without the need for adjustments.

TABLE I: Summary statistics

	value	gas used	gas price
mean	48.12	127511.76	3.01e+10
std	1495.49	239571.52	1.34e+13
min	0.00	21000.00	0.00
25%	0.06	40000.00	2.00e+10
50%	0.89	90000.00	2.00e+10
75%	1.11	90000.00	2.50e+10
max	1000000.00	4712388.00	3.62e+16

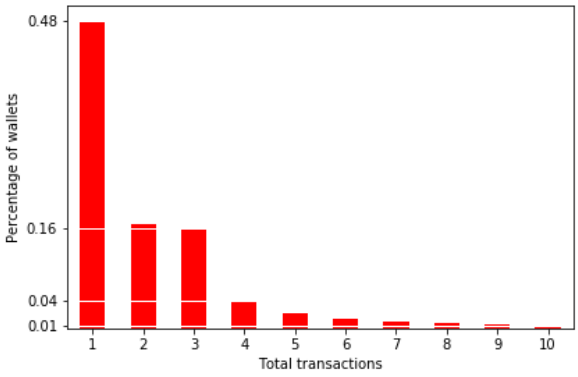
The initial student credit was enough to deploying the data-scraper on the *Microsoft Azure* servers for about a week. During this week we were able to extract 2 GB of data that consist of one million blocks with a total of 7.3 million transactions made from 415000 distinct wallets. Table I describes some rudimentary statistics.

Initially we also planned to scrape the IP-addresses of a wallet by making use of `web3.py`'s functionality which is provided for the `Bitcoin` blockchain. However, within the scraped dataset we were not able to identify a single address. The reason might that the `web3.py` module does not provide that functionality for the Ethereum blockchain in its current version. Mind you that we do not claim that it is impossible to retrieve the IP-address of a wallet. One proposed heuristic to do so is to assume that the first node that propagates a transaction through the network is also the one that is initializing the transaction (Reid & Harrigan, 2013). However, for us this would have meant to rewrite the data-scraper from `Python` into the `Go` language, which we considered to be out of this analysis' scope.

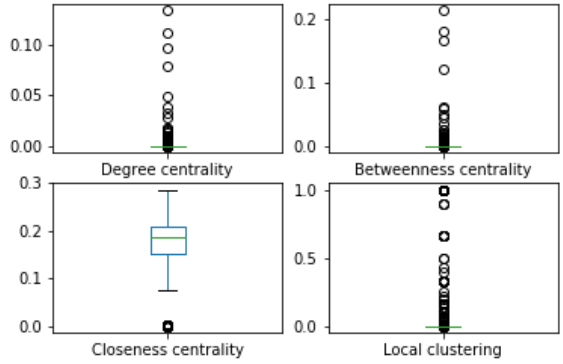
B. Descriptive Analytics

C. Graph Analysis

Looking at the rough outline of the observed period revealed that almost half of the wallets take part in one transaction during that time frame. Roughly 16 percent of the wallets appear two or three times respectively.



(a) Distribution of total number of transactions



(b) Centrality Distribution

Since the vast majority of accounts only seem to appear once inside the whole dataset we concluded that these are not interesting from a graph analysis perspective. Therefore we opted to use a subset from that data of accounts, where we could verify that they are interacting with other accounts. This subset consists of roughly four and a half million transactions in which 27.000 distinct wallets were either the sender or the receiver. From this subset we opted to create a graph in which the wallets are represented by the graph's nodes and the transactions are symbolized by the graph's edges.

Even this subset proved to be too large to handle, which means for the further graph-analysis that we were forced to rely on a random subset. However, if this random sample is large enough it should still give us an insight into the structure of the Ethereum network and the graph formed by this subset should still provide interpretable results.

Figure 3 displays the graph built from a random subsample of 1000 transactions, where again the nodes represent the different wallets and the edges stand for an observed transaction between them. The size of the node stands for the transaction's value. We can clearly see that most of the accounts form connections that consist of only two wallets and it is highly likely that these are also the accounts which are only observed once in the entire dataset.

Even here it is possible to see that there are subclusters of highly connected nodes associated with a high transaction value located in the middle of the graph. This appears despite the small sample size, which has to mean that these nodes in the center of the graph are important actors inside the network whose position will most likely become more pronounced the larger we choose the sample size to be. From this first glimpse at the network we would expect to find two tendencies inside the dataset. First, we expect a large number of nodes which are associated with low centrality measures and clustering coefficients. Secondly we expect to find some collections of nodes that form subgroups and within these subgroups we should be able to locate some nodes that play a tangible role in their respective local neighborhoods. Both of those observations do fit into the notion of a random graph, where we would expect a large number of loosely connected nodes with a low degree and an exponentially lower quanity of nodes which have a higher degree and who play a more central role inside the network.

Note nonetheless, that figure 3 provides only a rough outline since it is only using one thousand out of the 7.3 million available transactions. We pick a comparatively small sample size for this graph in order to avoid overplotting, the further analysis however uses a larger subsample of 100000 transactions.

The results of the larger subsample are summarized in table II and figure 2b. For the Degree Centrality we can observe a highly skewed distribution, where the mean lies at $\bar{C_D} = 0.0001$, whereas the first three quartiles remain firmely at 0. This indeed confirmes our previous suspicion that most nodes inside this network are merely loosely connected, even after removing the most isolated and remote nodes of the initial dataset. This means that most wallets interact only with are very limited set of other wallets and that it is highly uncommon for a node to accept transactions from many different wallets.

When looking at the results for the graph's Betweenness-Centrality, we can see a similiar pattern. The mean lies at $\bar{C_B} = 0.0001$ and all three quartiles are, again, 0. This means that the Betweenness centrality's distribution is also heavily skewed, which indicates that only a few nodes act as a mediator for their neighbors. This should not come as a surprise, since most of the nodes have a very low degree to begin with.

The Closeness centrality between all nodes paints a different picture. Its distribution is still skewed, however it is not as extremely skewed as the measures that have been discussed so far. The mean lies at $\bar{C_C} = 0.15$ with a standard deviation of $s_{C_C} = 0.078$. Note that this measure needs to be interpreted with care. Most of the nodes are disconnected, and thus do not have a connecting path at all. If a node j cannot be reached from a different node k there exist two common propositions in the literature. Either $d(j,k) = 0$ or $d(j,k) = \infty$. Most implementations rely on the former convention rather than the latter, since it avoids values that cannot interpreted as a

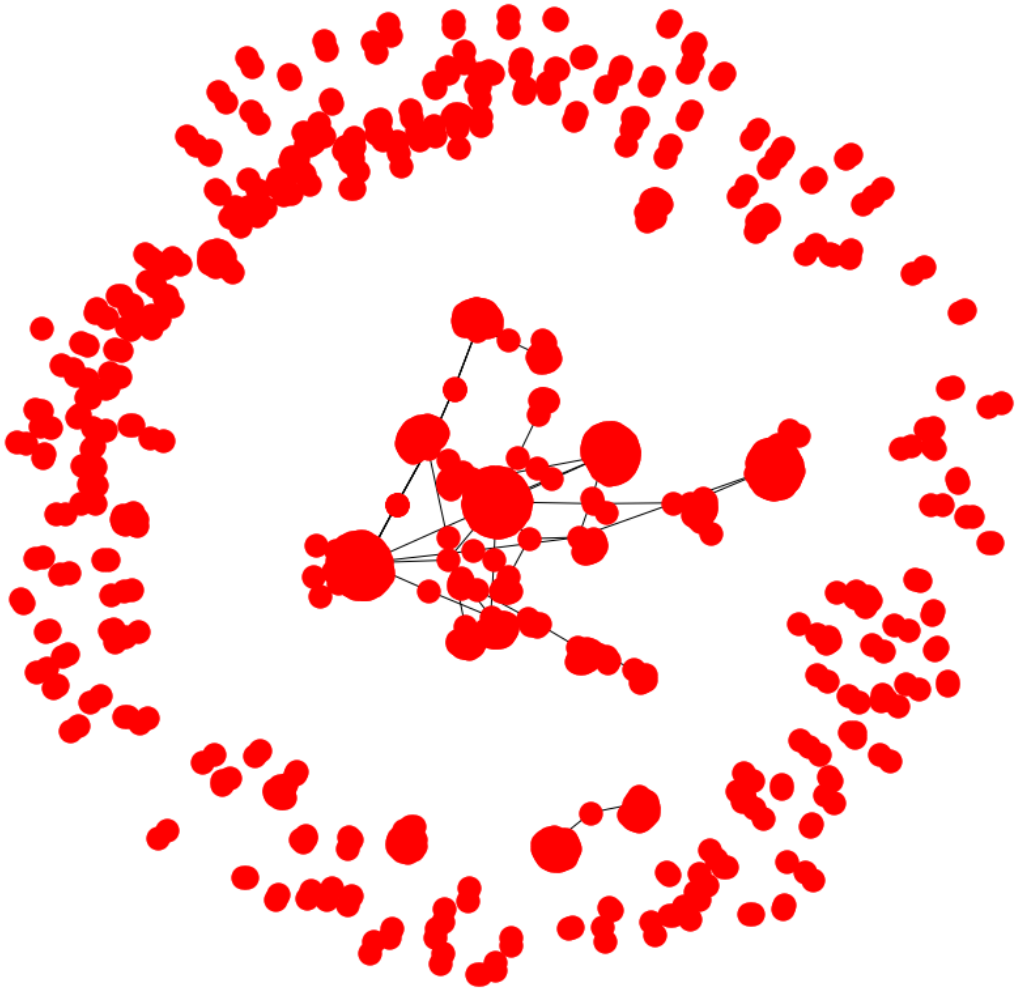


Fig. 3: Graph formed by a subsample of 1000 randomly drawn wallets

TABLE II: Centrality Distribution

	Degree centrality	Betweenness centrality	Closeness centrality	Local clustering
mean	0.0001	0.0001	0.1540	0.0043
std	0.0012	0.0020	0.0782	0.0636
min	0.0000	0.0000	0.0000	0.0000
25%	0.0000	0.0000	0.1522	0.0000
50%	0.0000	0.0000	0.1876	0.0000
75%	0.0000	0.0000	0.2072	0.0000
max	0.1344	0.2149	0.2862	1.0000

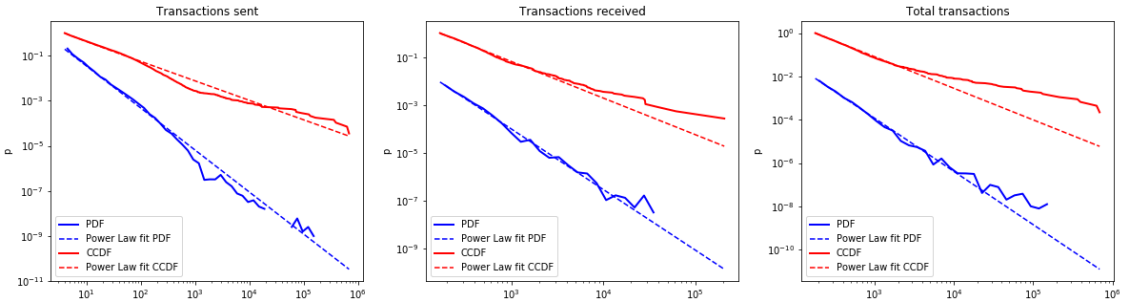


Fig. 4: Comparison with random graphs

number. This is also the convention that is implemented in the `networkx` module for the `Python` programming language (Hagberg, Schult, & Swart, 2008). But this also means that the closeness centrality should be interpreted as a local measure since the distances in the normalizing factor in (2), which are calculated as

$$\sum_{i=1}^{g} d(n,i),$$

will contain numerous zero-valued summands. So actually, this measure only makes sense for nodes that are of a degree k that is larger than three, since nodes of a lesser degree are necessarily close to each other.

The Local clustering coefficient accross the nodes is again highly skewed to the left. The mean lies at $\bar{C_L} = 0.004$ at a standard deviation of $s_{C_L} = 0.063$. Again the quartiles are all planted at 0. Note however, that there exist some nodes whose Local clustering coefficient reaches as far as $C_L(n) = 1$. This means that there are some nodes who are fully connected within their local cluster, which means that all triangles within that clique are centered around that node n. This, in turn, means that there are

So far all these findings indicate that the model of a random graph would be a good approximation of the real world data. In order to test this assumption in a graphical manner, we would like to compare the empirical density functions of the degrees against the theoretical powerlaw distribution. Remember that, in a random graph, the random variable $K(n)$ that describes the degree of a node n follows a powerlaw distribution

$$P(K(n) = k) \propto k^{-\alpha},$$

where α is some constant that has to be determined by Maximum-Likelihood methods (Meghanathan, 2018). Figure 4 presents the results for the empirical probability density function (PDF) and the complementary cumulative distribution function (CCDF). The CCDF is primarily used in the domain of signal processing and measures the the amount of time a signal spends above the mean power level of the measured signal. Equivalently, it assesses the probability that the signal power will be above the average power level (Udaigiriya & Sharma, 2015).

In order to reflect the difference in the roles accross the network such as the reciever or the sender of a transaction, we split the analysis into three different parts. First, we look at the distribution of the directed graph that is focusing on the actors that iniatate a transactions, i.e. the senders. Second, we look at the network formed by the receivers of a transaction and finally we look at the whole undirected graph that contains both receivers and senders. The results are displayed in figure 4. The PDF is shown by the blue line and the CCDF is represented by the red line. A high overlap between the dashed and the solid line indicates a good fit of the empirical distribution compared to the theoretical. Thus the probability density function of a perfectly random network would lie on its respective dashed line.

In the leftmost part of figure 4 we can see for the senders that the tails of the empirical PDF are closely following the theoretical distribution of the random graph. However, in the central part we observe one large deviation from the distribution of a random network. In particular, this means that - under a random graph - we would have expected more nodes that have a middling degree, i.e. $k = 3$, $k = 4$, and so forth. Instead, we observe a large number of loosely connected nodes.

The same holds for the sender's CCDF. The lower tail of the distribution follows the random graph closely, whereas the central part is too low, and the upper tail is too high for a random graph.

Looking at the wallets at the receiving end of a transaction, in the middle part of figure 4, we can discern a similar pattern. Both, the line for the PDF, as well as the line for the CCDF begin by closely following the theoretical distribution and begin to deviate after passing the median. Again we observe a density at the lower tail that is very similar to what we would expect from a random graph.

However, the most clear indicator of that pattern is shown for the composite graph, that contains both senders and receivers, in the rightmost part of figure 4. Here we observe a very pronounced deviation from the course of both the theoretical PDF and CCDF after reaching the median. Again, this is a clear indicator that there are too many nodes with a comparatively high degree, for this to be a random network.

VII. Conclusion

References

Alstott, J., Bullmore, E., & Plenz, D. (2014, 01). powerlaw: A python package for analysis of heavy-tailed distributions. *PloS one*, *9*, e85777. doi: 10.1371/journal.pone.0085777

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using networkx. In G. Varoquaux, T. Vaught, & J. Millman (Eds.), *Proceedings of the 7th python in science conference* (p. 11 - 15). Pasadena, CA USA.

Knorr, C. (2019, 01). Holland/leinhardt (1971): Transitivity in structural models of small groups. In (p. 267-270).

Meghanathan, N. (2018). Graph theoretic approaches for analyzing large-scale social networks. In (p. 7-34).

Newman, M. E. J. (2003). The structure and function of complex networks. *SIAM Review*, *45*, 167-256.

Piper, M., & Caver, J. (2018). *Web3py: A Python interface for interacting with the ethereum blockchain and ecosystem.* Retrieved from "https://github.com/ethereum/web3.py" ([Online; accessed 10-03-2019])

Reid, F., & Harrigan, M. (2013). An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks* (pp. 197–223). Springer.

Rossum, G. (1995). *Python reference manual* (Tech. Rep.). Amsterdam, The Netherlands, The Netherlands.

Udaigiriya, L., & Sharma, S. K. (2015). Complementary cumulative distribution function for performanceanalysis of ofdm signals and implement papr reduction techniques in ofdm system usinng ccdf function on matlab..

Watts, D., & H. Strogatz, S. (1998, 07). Collective dynamics of small world networks. *Nature*, *393*, 440-2. doi: 10.1038/30918