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.

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 V-A summarizes the relations within that database.

Algorithm 1 Data Scraper: Overview

```
Check progress file if the ETL can be resumed from a block.
if yes then
    current block $c \leftarrow$ content of progress file
else
    $c \leftarrow 1$
end if
$s \leftarrow 2000000$
while $c \leq s$ do
    Overwrite progress file with $c$
    $content_{Block} \leftarrow$ getBlock(c)
    $row_{Block} \leftarrow$ extractInfo($content_{Block}$)
    write($row_{Blocks}$, "Block")
    for each $transaction$ in $content_{Block}$ do
        $row_{Transaction} \leftarrow$ extractInfo($transaction$)
        write($row_{Transaction}$, "Transactions")
    end for
end while
```

B. Graph Analysis

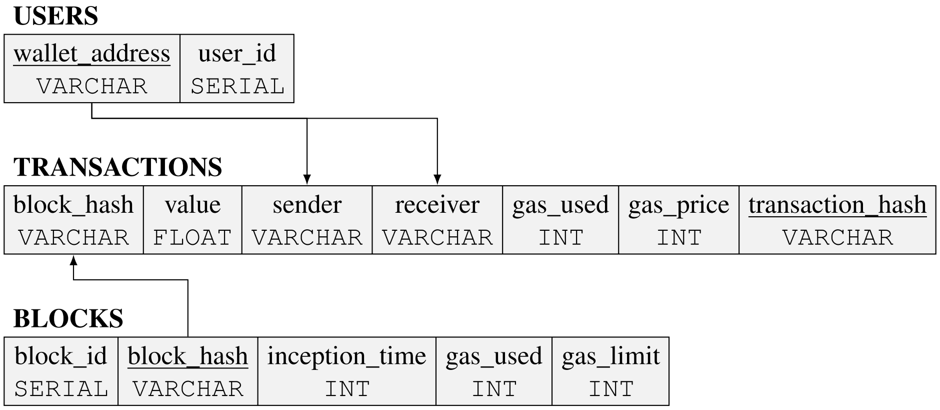
We can characterize a network by a number of metrics. The degree of a node in the network is the number of incoming or outgoing edges. We define a *triplet* i, v, j as an ordered set of three nodes, where v is the focal point and the undirected edges $< i, v >$ and $< v, j >$ form the neighboring edges. 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 edges in the shortest path between two nodes. The Closeness Centrality of a network with g nodes is defined as



USERS

wallet_address	user_id
VARCHAR	SERIAL

TRANSACTIONS

block_hash	value	sender	receiver	gas_used	gas_price	transaction_hash
VARCHAR	FLOAT	VARCHAR	VARCHAR	INT	INT	VARCHAR

BLOCKS

block_id	block_hash	inception_time	gas_used	gas_limit
SERIAL	VARCHAR	INT	INT	INT

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.

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

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:

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

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	3.016605e+10
std	1495.49	239571.52	1.341832e+13
min	0.00	21000.00	0.000000e+00
25%	0.06	40000.00	2.000000e+10
50%	0.89	90000.00	2.000000e+10
75%	1.11	90000.00	2.500000e+10
max	1000000.00	4712388.00	3.626777e+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.

B. *Descriptive Analytics*

C. *Graph Analysis*

Looking at the rou

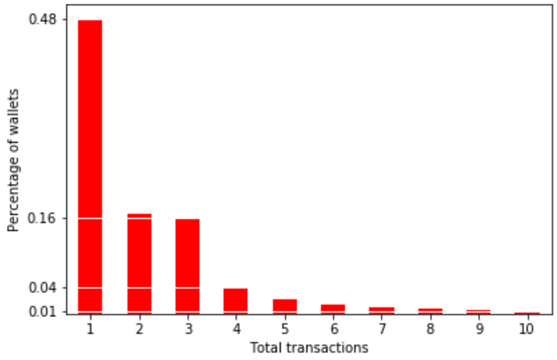


Fig. 2. Distribution of total number of transactions

VII. Conclusion

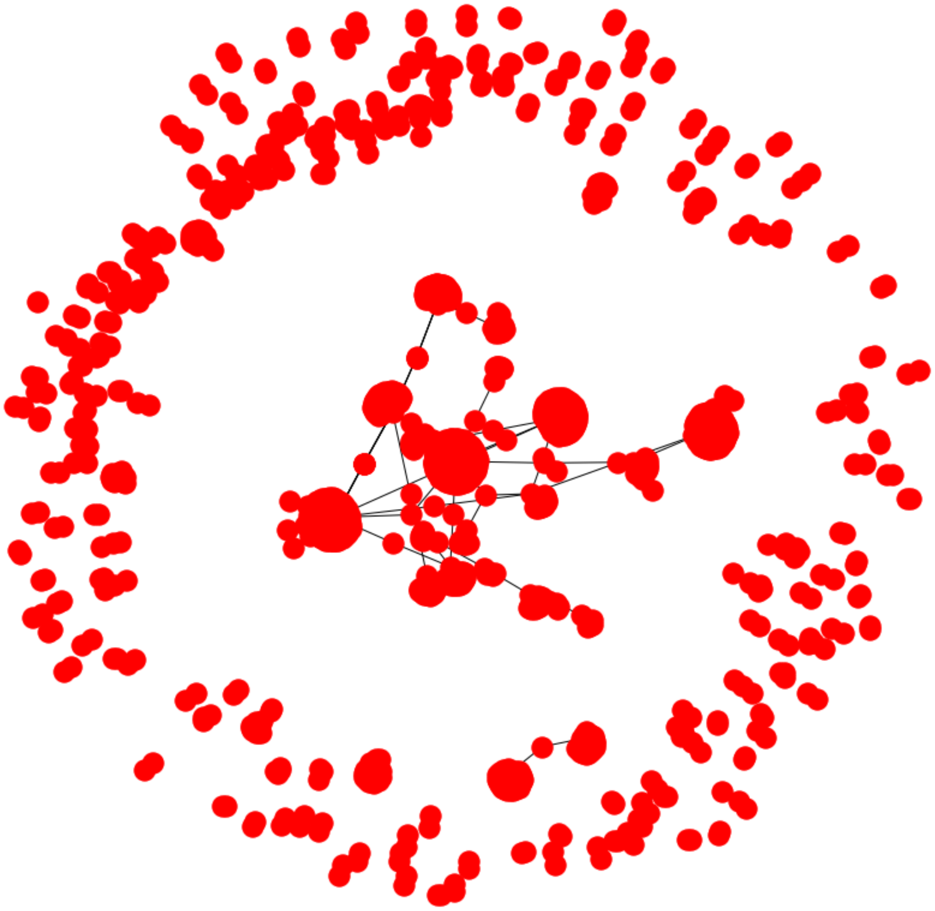


Fig. 3. Graph formed by a subsample of 1000 randomly drawn wallets

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

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

Lischke, M., & Fabian, B. (2016). Analyzing the bitcoin network: The first four years. *Future Internet*, *8*(1), 7.

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])

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

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

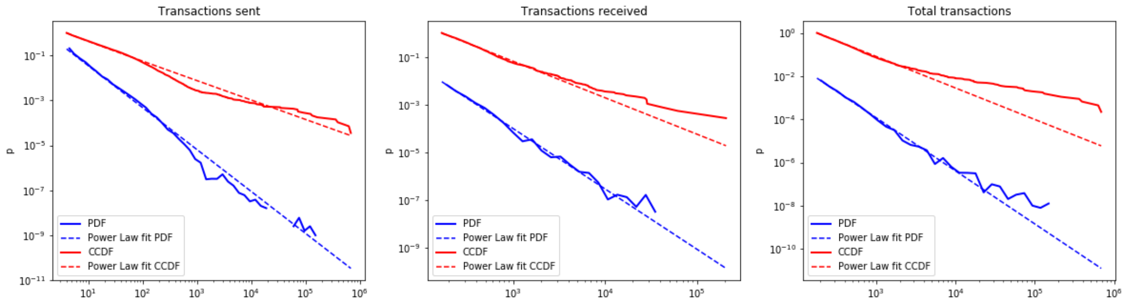


Fig. 4. Comparison with random graphs

10.1038/30918