Blockchain Anonymity and a Local Clustering Algorithm on Graph

Sung-Shine Lee

August 2018

1 Background

Since Satoshi proposed Bitcoin[4], the underlying technology blockchain, a decentralized public ledger, has demonstrated its capability of being a valid currency and its potential to shape future industries with its decentralized paradigm. The blockchain is maintained by trustless participants called miners running a distributed consensus protocol. For the system to be secure, it is required that no adversary can control a significant portion of the resource (such as computational power[4] or crypto assets[2] defined by the protocol). When running the protocol, miners will not only collect transaction information but also execute scripts that are specified in the transaction and update the blockchain accordingly.

The system can be viewed as a service that offers integrity and availability in a decentralized fashion. However, Bitcoin, often being misunderstood as an anonymous payment system, does not provide confidentiality due to its design nature. Specifically, to make a transaction, one has to broadcast all information about the transaction including the sender, the receiver, the amount, and other metadata. The information is then collected by miners who will publish it permanently on the blockchain. In this case, pseudonymity is offered by the system since accounts are represented by public keys thus not directly linked to the user identity. However, the whole transaction history of a given account is visible to the public.

While a user can have multiple accounts, it was shown by [5] to be possible to link the accounts under the same identity by analyzing their transaction history. Moreover, by matching on-chain transaction history with off-chain data, an adversary can connect accounts to the person in the real world[1]. Blockchain analysis is a tool that could potentially help identify criminals that use cryptocurrency as a tool to avoid tracking.

2 Blockchain Anlaysis as a graph

Every block of a blockchain contains information about multiple transactions. Multiple inputs and multiple outputs are recorded in a transaction and could technically belong to different people. However, a transaction is broadcasted by a single entity in the network, therefore it is often the case that the inputs in a transaction belong to the same person. The information that is being recorded on the blockchain can be used to construct graphs.

From the public information recorded in the blockchain, we could construct two types of graph intuitively: The transaction graph and the user graph. In a transaction graph, each vertex represents a transaction and each directed edge represents that the output of the source transaction is the input of the target transaction. In a user graph, each vertex represents a public key and each directed edge represents a transaction between the two public keys. Blockchain analysis tries to use different heuristics to identify public keys that belongs to the same user.

In other words, the goal is to cluster vertices on the user graph so that each cluster belongs to one single user. As the number of active public key in bitcoin is currently 500k, an algorithm that clusters vertices on a massive graph seems to be suitable for the task. The user graph connects public keys via payments, the connections between vertices are similar to those in the social network. It has been analyzed in [3] that the diameter of the graph is a large constant compared to graphs in the social network, however, the average distance between vertices is between 4 to 5. This indicates that although long paths exists in the graph, most of the paths are very short.

3 Local clustering algorithm for massive graphs

Spielman et al. [6] proposed a local clustering algorithm for massive graphs that executes in nearly linear time in respect to the cluster size it outputs. While processing the whole graph to produce a solution is often more accurate, the time needed for processing the whole graph is much larger and such algorithms are not suitable for graphs that are constantly evolving.

A local algorithm is an algorithm that takes a vertex as its input and examine only the vertices next to the ones that it has seen before in each step. Nibble, the algorithm proposed, can return a cluster within the graph G that has smaller conductance than the input parameter ϕ if it exists with probability at least $\frac{1}{2}$.

Performing random walk on a vertex will let the number of vertices with nonzero probability grow very fast, hence making it very hard to compute. The idea of Nibble is to ignore vertices with small probabilities to save computation and is able to maintain a distribution that is slightly larger than the cluster it needs to produce.

The random walk in Nibble will either stay at the same vertex for probability 1/2 or will move on to a neighboring vertex of the current one. The transition can be represnted by a matrix $M = ((AD^{-1} + I)/2$, where A is the adjacency matrix and D is the diagonal matrix with each element represents the degree of the corresponding vertex. Let the current distribution be x, then we could calculate the distribution of the following time step to be Mx. Following Nibble, we will truncate the small probability in the distribution by defining an ϵ and let all the vertices q with nonzero probabilities in Mx that are smaller than $\epsilon degree(q)$ to be zero.

```
C = Nibble(G, v, $\phi$, b).
    G is the graph, v is a vertex,
    0 < $\phi$ < 1, b is a positive integer.

    1. Set $\epsilon$ according to the graph G and b
    2. Initialize distribution vector that
       starts from vertex v
    3. Loop until "cluster found" or "time limit"
        (a) Calculate distribution of next timestep
              $r_t = Mr_{t-1}$
        (b) Perform distribution truncation
              $r_t = truncate_\epsilon(r_t)$
        (c) Check if there exist a cluster that satisfies
        the requirements.
```

The running time of Nibble is bounded by $O(\frac{2^b log^6 m}{\phi^4})$, where m is the number of edges in the graph G.

4 Discussions and open problems

While the algorithm mentioned in the paper is for undirected graph and the blockchain user graph is by nature a directed graph. Although one trivial way to apply the algorithm in the paper is to remove the direction in the original user graph, this may cause the new graph to lose information that is useful for analysis. Current blockchain analysis converts the graph into undirected and uses k-means algorithm to perform clustering, considering this:

1. How well will the clustering algorithm perform as a tool for blockchain analysis? Currently the heuristics of blockchain analysis directly cluster public keys that appear in the same input/output of a transaction and assert that they belong to the same user. However, by the construction of the user graph, the public keys that are being clustered in the heuristic does not necessarily have edges between the vertices.

2. Is there a better way to construct an undirected user graph to apply the algorithm? This problem comes naturally after the discussion of the previous one. Network and temporal information are often incorporated during blockchain analysis, it would be interesting to think of how to construct

a graph that incorporates the information naturally into a topological structure.

3. Is it possible to design $logm$ clustering algorithm for directed graphs?

On the bitcoin network, one way that has been developed to make blockchain analysis difficult is through mixing: a service that redistributes the coins among multiple users with unlinkable pattern. The service will first receive the fund from a user and mix the coins with those from others; It will then split the fund into several transactions that returns it to different addresses that belongs to the user. The anonymity of the service relies on the number of users and bitcoins that are mixing.

These mixing services behave differently than other nodes and is very likely to have interesting topology structure on the graph.

1. Currently the analysis around mixer is by analyzing publicly available mixing services. Is it possible to analyze the graph structure and identify private mixing services? I suspect that if we run local clustering algorithm on different users of a mixing services, the result will likely to include the node of the service.

2. As we could use clustering algorithm to identify public keys of the same user. Can we evaluate how well a mixing service perform by using the conductance ϕ?

Beside these, it may be crucial to compare the efficiency of traditional blockchain analysis and the ones of using local clustering algorithm.

5 Conclusion

In this paper, we've seen the connection between the blockchain users and graph structures; The connection between the clustering problem and blockchain analysis; and possible direction to research that help apply the local clustering algorithm.

As the blockchain is growing larger, it is important to recognize the efficiency of using local graph algorithms. The fact that the graph is dynamic and growing each second makes the problem even more significant.

References

[1] Steven Goldfeder, Harry Kalodner, Dillon Reisman, and Arvind Narayanan. When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies. *arXiv preprint arXiv:1708.04748*, 2017.

[2] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19, 2012.

[3] Damiano Di Francesco Maesa, Andrea Marino, and Laura Ricci. Data-driven analysis of bitcoin properties: exploiting the users graph. *International Journal of Data Science and Analytics*, pages 1–18, 2017.

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

[5] Dorit Ron and Adi Shamir. Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security*, pages 6–24. Springer, 2013.

[6] Daniel A Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on Computing*, 42(1):1–26, 2013.